```
<!DOCTYPE html>

<!-- In this exercise, we have a textbox in a form that, when submitted, inserts the
    text in the textbox into a div -->

<html>
    <head>
        <title>Exercise 1!!!!!!</title>
        <meta charset="UTF-8">

        <!-- Include jquery (link to Google's hosted version so we don't have to download locally -->
        <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.1/jquery.min.js"></script>

        <!-- Remember that you can extract this javascript into a separate .js file. You will
            probably want to do that for the first project, since it's cleaner -->
        <script>
            // Remember that the dollar sign almost always means something jquery-related.
            // Here, we are saying that the function we are passing as the second argument
            // should be run as soon as the "ready" event is triggered on the whole document.
            // The ready event is triggered as soon as the page finishes loading completely.
            //
            // The next comment explains *why* we need to do wait until the page finishes
            // loading before doing everything else.

            $(document).on('ready', function() {

                // Here, we are binding a function to the "submit" event on the element with id
                // exercise1-form. The submit event is triggered when you try to submit that
                // form. The bound function is run when the submit event is triggered.
                // Here, like with the document-ready event handler above, we used an anonymous
                // function as the second argument, but note that you could also define a completely
                // separate function and pass it to the "on" function by name.
                //
                // So why did we need to wait for the document to be ready before executing this line
                // of code? If we didn't wait, then at this point in time the form with id "exercise1-form"
                // wouldn't have been loaded yet, and so we wouldn't be able to bind this anonymous
                // function to the non-existant form's submit event.
                $("#exercise1-form").on('submit', function() {

                    // Here, we select the element with the id "exercise1-text", which is our text input
                    // box, and get the text that's currently inside it using the val() function (see the
                    // jquery documentation). If instead of getting the contents of the text box, you wanted
                    // to *set* the contents, you would pass an argument to val(). For example,
                    // $("#exercise1-text").val("");
                    // would remove all text from the text box.
                    //
                    // Note that if I didn't say "var" here, we would be declaring the variable textbox
                    // as a global variable, which is generally bad style.
```

```
                    var textbox = $("#exercise1-text").val();

                    // Here is an alternative of the above line, using pure javascript (e.g., no jquery):
                    // var textbox = document.getElementById('exercise1-text').value;

                    // Here, we grab the element with id "exercise1-div", and set the text inside it
                    // to what is currently inside of the "textbox" variable. A similar function to the
                    // text() function is the html() function -- check the jquery documentation to be
                    // sure you understand the differences (and the consequences of using html());.
                    //
                    // Here, we are setting the text contents of the div using the text() function. But,
                    // like the val() function, if instead we wanted to *get* the text contents, we would
                    // call text() without any arguments.
                    $("#exercise1-div").text(textbox);

                    // We need to return false in order to prevent the default behavior of the "submit"
                    // event on a form, which is, of course, actually submitting the form. If we didn't
                    // return false, you would see the page refresh.
                    return false;
                });
            });
        </script>
    </head>
    <body>
        <form id="exercise1-form">
            <input type="text" id="exercise1-text">
            <input type="submit">
        </form>
        <!-- We in-lined the CSS here, but for your project note that inline CSS is generally bad style! -->
        <div id="exercise1-div" style="border:1px solid red">
            Div!!
        </div>
    </body>
</html>
```

```
<!DOCTYPE html>

<!-- We expand on exercise 1, where now the textbox should take a zip code, and we
    insert into the div the city and state corresponding to that zip code -->

<html>
    <head>
        <title>Exercise 2!!!!!!!!!!!</title>
        <meta charset="UTF-8">
        <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.1/jquery.min.js"></script>
        <script>
            $(document).on('ready', function() {
                $("#exercise2-form").on('submit', function() {

                    // We don't validate that this is a zipcode, but it might be a good idea to do so.
                    var zipcode = $("#zipcode").val();

                    // Everything above this point, save some name changes, is the same as
                    // in exercise1.html.

                    // Here, we construct the appropriate URL for our geocoding request to Google's API.
                    // Remember that we can use "+" to concatenate variables, such as zipcode, to strings.
                    var url = "http://maps.googleapis.com/maps/api/geocode/json?address=" + zipcode + "&sensor=true";

                    // Check the jquery documentation for how the getJSON() function works. Remember that it
                    // just uses the ajax() function underneath the hood, and jquery just provides this
                    // function as a convenience.
                    //
                    // The first argument is the url we want to grab the JSON from, and the second is a function whose
                    // first argument is the returned JSON. Be sure to understand this asynchronous aspect of getJSON
                    // (and AJAX in general).
                    //
                    // Also remember to handle all possible errors in your projects! Are we handling all of them
                    // here?
                    $.getJSON(url, function(data) {

                       // From the Google geocoding documentation, we see that data.status might indicate that
                       // something went wrong. Why might zero results be returned by Google?
                       if (data.status === "ZERO_RESULTS") {
                           alert('uh oh');
                           return;
                       }

                       // Each element in "components" is an object representing a specific "area level",
                       // e.g., "Cambridge", "Massachusetts", "USA", etc.
                       var components = data.results[0].address_components;

                       // Iterate over these address components, looking for the components representing
```

```
                        // city and state.
                        for (var i = 0; i < components.length; i++) {

                            // If the "types" array contains "locality", we know that this address
                            // component represents the city. indexOf only returns -1 if the given
                            // argument can't be found in the array.
                            if (components[i].types.indexOf("locality") !== -1) {
                                var city = components[i].long_name;
                            }
                            // If the "types" array contains "administrative_area_level_1", we know
                            // that this address component represents the city.
                            else if (components[i].types.indexOf("administrative_area_level_1") !== -1) {
                                var state = components[i].short_name;
                            }
                        }
                        // Remember that the "typeof foo === 'undefined'" pattern is a way of checking to
                        // see if variable foo hasn't been defined.
                        //
                        // Here, if city or state is undefined, then we didn't find them in the for loop
                        // above, and so can't print them as desired.
                        if (typeof city === 'undefined' || typeof state === 'undefined') {
                            alert('uh oh');
                            return;
                        }

                         // Insert the city and state into the div with id "city-state"
                         // Again, we use "+" to concatenate the variables.
                         $("#city-state").text(city + ', ' + state);

                    });

                    // As in the previous exercise, prevent the form from actually submitting.
                    // Notice this is *outside* the anonymous function passed to getJSON.
                    return false;
                });
            });
        </script>
    </head>
    <body>
        <form id="exercise2-form">
            <input type="text" id="zipcode">
            <input type="submit">
        </form>
        <div id="city-state" style="border:1px solid red">
        </div>
    </body>
</html>
```

```
<!DOCTYPE html>

<!-- We expand on exercise 2, where now we just use the user's current location instead
    of asking for the zip code, and insert into the div the city and state corresponding to
    the user's current location -->

<html>
    <head>
        <title>Exercise 3!!!!!!!!!!!!!!!!</title>
        <meta charset="UTF-8">
        <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.1/jquery.min.js"></script>
        <script>
            $(document).on('ready', function() {

                // Everything above this point is the same as in the prior exercises.

                // Notice here we've changed the event to which we're binding from "submit" to "click". Why's that?
                // We are no longer using a form (which you *submit*), but now a button (which you *click*). See
                // the HTML below to see that the element with id "geolocate-button" is in fact a button.
                $("#geolocate-button").on('click', function() {

                    // Use geolocation to grab the user's current location. This position object is passed as an
                    // argument to the anonymous function passed to getCurrentPosition.
                    //
                    // Note that older browsers won't have a "navigator" object available. What can you do to prevent
                    // this causing an error?
                    //
                    // Also, look at the documentation for navigator.geolocation.getCurrentPosition, and see that it
                    // takes an optional second argument for handling error conditions. When might errors arise? You
                    // will probably want to handle those errors in your project!
                    navigator.geolocation.getCurrentPosition(function(position) {
                        var lat = position.coords.latitude;
                        var lng = position.coords.longitude;

                        // Notice that, compared to the previous exercise, the URL has now changed such that the get
                        // parameter is called "latlng" instead of "address", as per the Google geocoding API's
                        // documentation.
                        var url = "http://maps.googleapis.com/maps/api/geocode/json?latlng=" + lat + ',' + lng + "&sensor=true";

                        // All javascript below this point is identical to the previous exercise.
                        $.getJSON(url, function(data) {

                           if (data.status === "ZERO_RESULTS") {
                               alert('uh oh');
                               return;
                           }
                           var components = data.results[0].address_components;

```

```
                            for (var i = 0; i < components.length; i++) {
                                var comp = components[i];

                                if (comp.types.indexOf("locality") !== -1) {
                                    var city = comp.long_name;
                                }
                                else if (comp.types.indexOf("administrative_area_level_1") !== -1) {
                                    var state = comp.short_name;
                                }
                            }

                            if (typeof city === 'undefined' || typeof state === 'undefined') {
                                alert('uh oh');
                                return;
                            }

                             $("#city-state").text(city + ', ' + state);

                        });
                    });

                    // As always, prevent the form from actually submitting.
                    return false;
                });
            });
        </script>
    </head>
    <body>
        <button id="geolocate-button">Use current location!</button>
        <div id="city-state" style="border:1px solid red">
        </div>
    </body>
</html>
```

```
<!DOCTYPE html>

<!-- We expand on exercise 1 (NOT exercise 3), where now we append to the div the textbox's
    current contents instead of replacing all of the contents, and also use localStorage
    to remember the contents of the div so that it's still there when we refresh the page -->

<html>
    <head>
        <title>Exercise 4!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!</title>
        <meta charset="UTF-8">
        <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.1/jquery.min.js"></script>
        <script>
            $(document).on('ready', function() {


                // We need this check in case we are in an older browser, in which case localStorage
                // wouldn't be defined.
                if (typeof localStorage !== 'undefined') {

                    // If this is the first time we are loading the page (or we've never entered text into the div
                    // before), we shouldn't try to insert localStorage.divtext (we could call it whatever we
                    // want) into the page, since its contents won't yet be defined.
                    if (typeof localStorage.divtext !== 'undefined') {
                        // The localStorage variable already exists, so insert its contents into
                        // the div. Notice we use html(), so that the <br> tags are preserved.
                        // What problems might this cause? Think about what would happen if
                        // we insert html into the text box. How could we prevent this?
                        //
                        // Also notice that this line is inside the "document-ready" event handler,
                        // so the element with id exercise4-div definitely already exists.
                        $("#exercise4-div").html(localStorage.divtext);
                    }
                }

                $("#exercise4-form").on('submit', function() {

                     var textbox = $("#exercise4-text").val();

                     // We change this from exercise 1 to use append() instead of text(), so we append to
                     // the div instead of replacing the contents.
                     $("#exercise4-div").append(textbox + '<br>')

                     // Grab *all* of the contents, including the html tags, inside of the div. Notice we
                     // use html() instead of text().
                     var alltext = $("#exercise4-div").html();

                     // Store the stuff from the div inside of localStorage.divtext, so we can access it after
                     // a page refresh.
```

```
                    localStorage.divtext = alltext;

                    // Prevent the form from submitting.
                    return false;
                });
            });
        </script>
    </head>
    <body>
        <form id="exercise4-form">
            <input type="text" id="exercise4-text">
            <input type="submit">
        </form>
        <div id="exercise4-div" style="border:1px solid red">
        </div>
    </body>
</html>
```