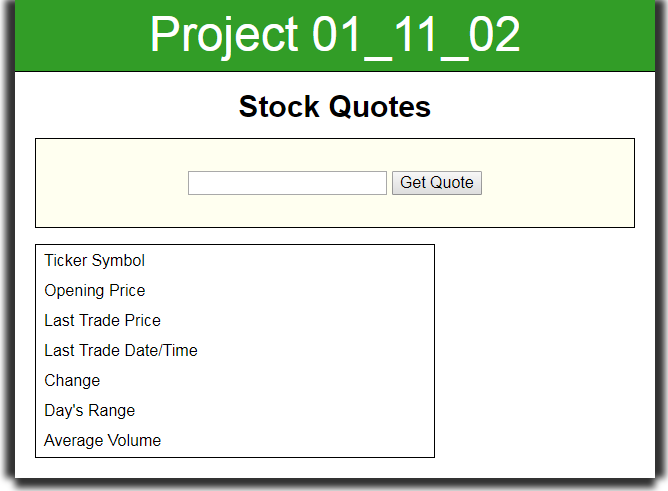
**Project 01\_11\_02**

In this project, you will create use a PHP script as a proxy to fetch market information for a stock based on a stock symbol that a user specifies.

1. Copy all of the completed files from ***Project 01\_11\_02.zip*** into a ***Project 01\_11\_02*** 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 JavaScript file ***script.js***. At the bottom of the file, add the ***global*** variable ***httpRequest*** and set its value to ***false***. Create a second global named ***entry*** with a text value of ***“^IXIC”***:  
   ***/\* global variables \*/  
   var httpRequest = false;  
   var entry = "^IXIC";***
3. Directly below that, create a function named ***getRequestObject()***. Fill the function with a ***try*** / ***catch*** structure to create an ***XHR*** object. Add an alert() that we will use to test the function:  
    ***try {  
    httpRequest = new XMLHttpRequest();  
    }  
    catch (requestError) {  
    return false;  
    }  
    alert(httpRequest);  
    return httpRequest;***
4. Let’s set up a test for this by setting up a temporary event handler for the page ***load*** event. At the bottom of the file, create the following code:  
   ***if (window.addEventListener) {  
    window.addEventListener("load", getRequestObject, false);  
   }   
   else if (window.attachEvent) {  
    window.attachEvent("onload", getRequestObject);  
   }***  
   Let’s move the code and do a server test. Page load should trigger an ***alert()*** showing a valid ***XHR*** object.
5. To stop any default submission from executing, let’s build a function to stop it below the ***getRequestObject()*** function, with an ***alert()*** test:  
   ***function stopSubmission(evt) {***  
    ***alert("stopSubmission()");  
    if (evt.preventDefault) {  
    evt.preventDefault();  
    }  
    else {  
    evt.returnValue = false;  
    }  
   }***
6. Let’s modify our event handler creation code to call that function as an event handler on the ***submit*** event:  
   ***var form = document.getElementsByTagName("form")[0];  
   if (form.addEventListener) {  
    form.addEventListener("submit", stopSubmission, false);  
    window.addEventListener("load", getRequestObject, false);  
   }   
   else if (form.attachEvent) {  
    form.attachEvent("onsubmit", stopSubmission);  
    window.attachEvent("onload", getRequestObject);  
   }***Let’s do a server test for syntax and to make sure that both our load event and submit event are acting correctly.
7. Now let’s start to build out a function that will request stock quote data from the server. It will use the stock symbol the user enters in the ***<input>*** field, or the default if nothing is entered. We will start it by having it create our XHR object if necessary, or re-use it if already created. Below ***stopSubmission()*** enter:  
   ***function getQuote() {  
    alert("getQuote()");  
    if (document.getElementsByTagName("input")[0].value) {  
    entry = document.getElementsByTagName("input")[0].value;  
    }  
    if (!httpRequest) {  
    httpRequest = getRequestObject();  
    }  
   }***
8. To test this, let’s first call the function at the bottom of our stopSubmission() function:  
    ***getQuote();***
9. Let’s also modify our ***load*** event to use it as an event handler instead of ***getRequestObject()***:  
   var form = document.getElementsByTagName("form")[0];  
   if (form.addEventListener) {  
    form.addEventListener("submit", stopSubmission, false);  
    window.addEventListener("load", ***getQuote***, false);  
   }   
   else if (form.attachEvent) {  
    form.attachEvent("onsubmit", stopSubmission);  
    window.attachEvent("onload", ***getQuote***);  
   }  
   Now we can do a server test on both the ***load*** event, which should produce a ***getQuote()*** alert and a valid ***XHR*** object alert. The ***submit*** event. Should produce a ***stopSubmission()*** alert and a ***getQuote()*** alert. If everything looks good let’s remove all of the debug alerts from the code and go on to conquer new ground.
10. We can now prepare our ***getQuote()*** function to generate our AJAX request:  
     ***httpRequest.abort();  
     httpRequest.open("get", "StockCheck.php?t=" + entry, true);  
     httpRequest.send(null);***A browser test with Developer Tools will be required to see some results. Go to the Network tab. We should see a Request URL that is properly formed and has our default stock symbol ***^IXIC*** attached. Now type in Microsoft’s ***MSFT*** and it should show that ***Submit*** produces the right results.
11. Now we need to get our data, so we need to set our onreadystatechange event handler:  
     ***httpRequest.onreadystatechange = displayData;***
12. Let’s start to build out the ***displayData()*** event handler below ***getQuote()***. First we will see if we can get data back, before we attempt to start placing that data on the page with the DOM:  
    ***function displayData() {***  
     ***if (httpRequest.readyState === 4 && httpRequest.status === 200) {  
     var stockResults = httpRequest.responseText;  
     console.log(stockResults);  
     }  
    }***  
    A server test with Developer Tools in the Console tab should let us test. Make sure there are no syntax errors. We should get good ***console.log()*** data. Now let’s try it with a stock symbol of ***MSFT***.
13. Now let’s start to format that data so that we can get it into the DOM and onto the page. At the bottom of the ***if*** statement, first remove the debug ***console.log()***. Now, let’s format the text string that we got in from ***responseText***. First, we will split that string into subarrays with a JavaScript ***Array.split()*** method and a regular expression that will split it based on the commas and quotes. We will ***console.log()*** the result:   
     ***var stockItems = stockResults.split(/,|\"/);  
     console.log(stockItems);***In the Console tab, expand the array. You will notice that it looks weird. We get ***empty strings*** as elements, mixed with the ***data*** converted to strings as elements. We purposely set up the regular expression to do that, to split on both commas and quotes. This has the effect of taking any quote it finds and making it into a string element, and taking the data inside it and enclosing it in quotes. Any data it finds not in quotes, it puts into quotes. So all the data is now strings, even the data that was numeric. This makes it easier for us to deal with to place into the DOM elements.
14. Now let’s get rid of those empty strings. To do this we will use a ***for*** loop with a JavaScript ***Array.splice()*** method. It specifies the array position to remove it from, and how many elements to remove. We will ***console.log()*** the result:  
     ***for (var i = stockItems.length - 1; i >= 0; i--) {  
     if (stockItems[i] === "") {  
     stockItems.splice(i, 1);  
     }  
     }  
     console.log(stockItems);***  
    In the Console tab, expand the new array. If everything is correct, we now have the empty strings removed, and the rest of the data is still there, set up as string elements in the modified array. If we are good to go, remove the three ***console.log()*** debug statements.
15. Now let’s see if we can get some data onto the page by placing it into the DOM nodes. We will start with a single element by placing the following code directly below the ***for*** loop:  
     ***document.getElementById("ticker").innerHTML = stockItems[0];***  
    A server test should give us the stock symbol in the right place.
16. Now let’s get the rest of the data up on the page by pulling the rest of the array elements and placing them in the DOM nodes:  
     ***document.getElementById("openingPrice").innerHTML =   
     stockItems[6];  
     document.getElementById("lastTrade").innerHTML =   
     stockItems[1];  
     document.getElementById("lastTradeDT").innerHTML =   
     stockItems[2] + ", " + stockItems[3];  
     document.getElementById("change").innerHTML = stockItems[4];  
     document.getElementById("range").innerHTML = (stockItems[8]   
     \* 1).toFixed(2) + " &ndash; " + (stockItems[7] \* 1).toFixed(2);  
     document.getElementById("volume").innerHTML = (stockItems[9]   
     \* 1).toLocaleString();***  
    The ***^IXIC*** symbol will give us data for the entire NASDAQ. If we try it for ***MSFT***, the data should make sense.
17. Now let’s get a little better style into the stock data. To do that, we will build a ***formatTable()*** function directly below ***displayData()***:  
    ***function formatTable() {***  
     ***var rows = document.getElementsByTagName("tr");  
     for (var i = 0; i < rows.length; i++) {  
     rows[i].style.background = "#9FE098";  
     }  
    }***
18. Now we need to add this as an event handler to be triggered by the ***load*** event. Add the following code to our event handler setup area:  
    if (form.addEventListener) {  
     form.addEventListener("submit", stopSubmission, false);  
     ***window.addEventListener("load", formatTable, false);***  
     window.addEventListener("load", getQuote, false);  
    }   
    else if (form.attachEvent) {  
     form.attachEvent("onsubmit", stopSubmission);  
     ***window.attachEvent("onload", formatTable);***  
     window.attachEvent("onload", getQuote);  
    }  
    Let’s give this a server test, and we should get a good effect.
19. Now let’s make sure our data is up to date when the site is active. We can do this with a setTimeout() code sequence to refresh every 10 seconds. We will use a console.log() debug statement to make sure it’s working. Add the following code to the top of ***getQuote()***:  
     ***console.log("getQuote()");***Add the following sequence to the bottom of getQuote():  
     ***clearTimeout(updateQuote);  
     var updateQuote = setTimeout('getQuote()', 10000);***  
    Let’s give this a server test. If it is working remove the ***console.log()*** debug and it is a wrap.