# Assignment08 – Lemonade Forecast

As you have seen in previous assignments, the weather forecast plays an important role in our lemonade stand business.

In this assignment, we will use an API to plot a simple chart to represent the expected noon-day temperature (in degrees Celsisus) for a week. You will also display the expected demand (in cups) for lemonade on each day.

You will use your knowledge of functions, strings, numbers, scope, Ajax and jQuery to write this code.

## Overview

Important notes:

* The URL for the API is here:

<https://misdemo.temple.edu/lemonweather/forecast>

* The API returns randomly generated data that simulates one week’s worth of rising and falling noonday temperatures measured in degrees Celsius.
* The API returns random, dynamically changing, data. You will get a different forecast every time you call the API.
* Take a moment to examine the JSON data returned by the API. It’s easiest to read the JSON if you view the URL using the Firefox browser. Take a few minutes to examine the structure of the data, and to identify what key value pairs you will have to work with. This small investment of time can save you a lot of time and aggravation later!
* You can see an illustration of what the solution should look like when you are done by watching the following YouTube video. (This video has no audio)

<https://youtu.be/POsCF961snM>

* This assignment will expect you to use three jQuery methods. You can read about them here.
  + The getJSON method – <https://www.w3schools.com/jquery/ajax_getjson.asp>
  + The html method – <https://www.w3schools.com/jquery/html_html.asp>
  + The append method -- <https://www.w3schools.com/jquery/html_append.asp>

## Instructions

1. Download assignment08.zip as provided by your instructor.
2. Start your work by completing the function makeString. This function takes one integer n as a parameter and returns a series of “X” characters. The number of “X” characters is determined by the value of n.

For example:

* + - makeString(1) returns “X”
    - makeString(2) returns “XX”
    - makeString(5) returns “XXXXX”
    - and so on..

***You do need a loop*** to complete the makeString function. This function ***does not need error trapping.***

1. Continue your work by completing the function getForecast. You ***do not need a loop*** to complete this function. (Although, you could try if you wanted to!)

**Here is the pseudo code:**

* 1. Write a jQuery getJSON statement that calls the forecast API
  2. Inside the callback function of the statement, console log the data returned by the API. This is not strictly necessary, but it is a best practice that aids both you the student and the instructor’s evaluation of your work.
  3. Continue to work inside the callback function.
     + Use the jQuery html method to put the string “Sun “ into the HTML div tag with the id of sun. Like this:

$("#sun").html("Sun ");

* + - Now append to the #sun tag a series of “X” characters. One X for every degree Celsius recorded for Sunday. Use the makeString function you already wrote to do this.
    - Now append to the #sun tag the forecasted number of cups of lemonade needed that day.
    - Now repeat this operation for Mon, Tue, Wed, Thu, Fri, and Sat.

1. Check your work. Clicking on the forecast button should cause new forecast information to appear.
2. Publish your work to misdemo. Check your URL.
3. Go to canvas and turn in your URL there. A correct URL will start with **https://misdemo** and end in **assignment08/forecast.html**

**CONTINUED**

## How will this assignment be graded?

|  |  |
| --- | --- |
| The function getString works as expected. | 40 |
| The function getForecast works as expected. (Your solution \*must\* use the API to retrieve dynamic data.) | 40 |
| Presentation is as demonstrated in the video | 10 |
| Solution was uploaded to misdemo correctly | 5 |
| Student provided a working / correct URL | 5 |
| **TOTAL** | **100** |