## 3.Forecaster

Write a program that requests a weather report **from a** **server** and **displays** it to the user.

Use the skeleton from the provided resources.

When the user writes the name of a location and clicks “Get Weather”, make a GET request to the server at address http://localhost:3030/jsonstore/forecaster/locations. The response will be an array of objects, with the following structure:

{

name: locationName,

code: locationCode

}

Find the object, corresponding to the name that the user submitted in the input field with ID "location" and use its **code** value to make **two more** GET **requests**:

* For current conditions, make a request to:

**http://localhost:3030/jsonstore/forecaster/today/:code**

The response from the server will be an object with the following structure:

**{**

**name: locationName,**

**forecast: { low: temp,**

**high: temp,**

**condition: condition }**

**}**

* For a 3-day forecast, make a request to:

**http://localhost:3030/jsonstore/forecaster/upcoming/:code**

The response from the server will be an object with the following structure:

**{**

**name: locationName,**

**forecast: [{ low: temp,**

**high: temp,**

**condition: condition }, … ]**

**}**

Use the information from these two objects to compose a forecast in HTML and insert it inside the page. Note that the <div> with ID "forecast" must be set to visible. See the examples for details.

If an **error** occurs (the server doesn’t respond or the location name cannot be found) or the data is not in the correct format, display "**Error**" in the **forecast section**.

Use the following codes for weather symbols:

* Sunny **&#x2600;** // ☀
* Partly sunny **&#x26C5;** // ⛅
* Overcast **&#x2601;** // ☁
* Rain **&#x2614;** // ☂
* Degrees **&#176;**  // °

**Examples**

When the app starts, the **forecast div** is **hidden**. When the user **enters a name** and **clicks** on the button **Get Weather**, the requests being.







