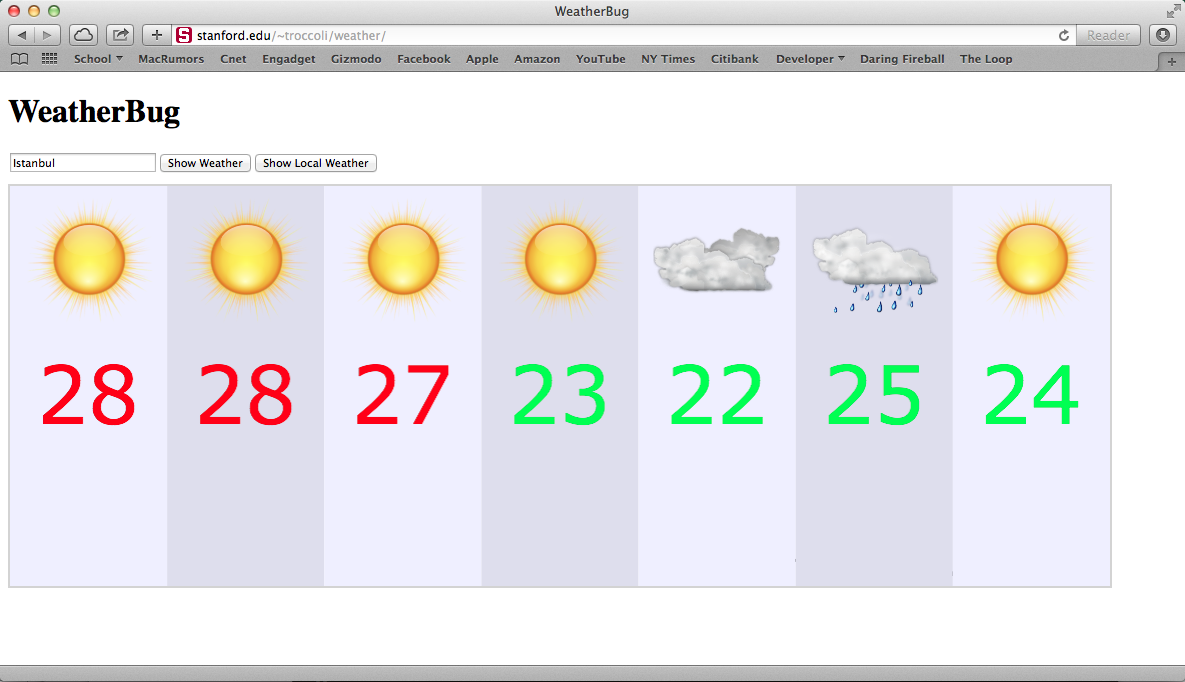
Stanford CS in Istanbul Summer 2014

**JavaScript: WeatherBug**



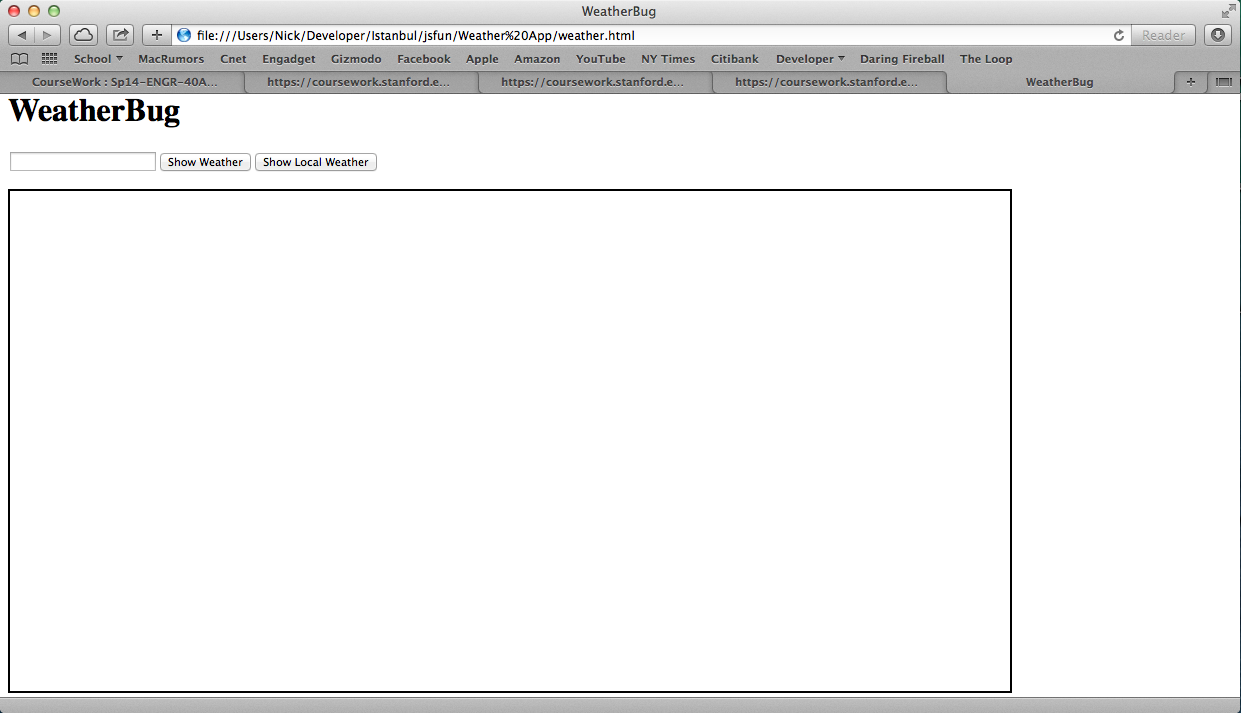
**The WeatherBug App**

If you’ve ever used weather.com or another weather site to look up the weather, you know how useful (and cool!) it is to be able to look up the weather online for both your current location and any location around the world. And, as it turns out, it’s also a great way to learn about and practice JavaScript callbacks! In this assignment, you’re going to complete a website that, using JavaScript, fetches and displays weather for any place the user types in, or the user’s current location.

We know you’re all GRect and GLabel ninjas – therefore, we’ve done all of the graphics code for you! What we want you to do is write all of the callbacks to make the website work properly. We suggest you break this down into milestones, and get each milestone working before moving on.

**Milestone 1: Buttons**

The first task you should tackle is getting your website’s button callbacks working properly. Your page should have two buttons – “Show Weather” and “Show Local Weather”.



To add buttons to the screen, use the addButton() method of GraphicsApp. addButton() takes 2 parameters – the text the button should display, and the function callback that should be executed when the button is clicked.

Remember that when you want to pass a function as a callback, you type the name of the function *without parentheses*.

**Milestone 2: The “Show Weather” Button**

Now that you have your buttons set up, it’s time to tackle the first of the two buttons – fetching weather for a place the user types in. You can get the text the user typed in the text field from the **value** property of your text field variable. Otherwise, there are only 2 **GraphicsApp** functions you’ll need for this milestone: fetchWeatherForQuery() and displayErrorMessage(). Your website should fetch the 7-day forecast for the place the user typed in. If the fetch is successful, you should display the weather data that you get back. Note that the success callback function should take one parameter – the weather data.

Remember, when fetching data from the web, the request could fail for a variety of reasons. That’s why you need to pass an error callback when fetching weather data. If the fetch does fail, you should display an error message to the user using displayStatusMessage().

**Milestone 3: The “Show Local Weather” Button**

Now let’s tackle the second button – fetching weather for the user’s current location. Instead of getting the text the user entered, you should instead fetch the user’s current location coordinates (latitude and longitude). Getting the current location requires a callback, because it takes a long time to locate the user! That callback function should take 2 parameters – the latitude and longitude.

Once you get the user’s location, your website should again fetch the 7-day forecast for the place the user typed in, but this time using fetchWeatherForCoodinates(). If the fetch is successful, you should display the weather data that you get back. Note that the success callback function should take one parameter – the weather data.

Again, remember that, when fetching data from the web, the request could always fail for a variety of reasons. That’s why you need to pass a second callback when fetching weather data. If the fetch does fail, you should display an error message to the user using displayStatusMessage().

**Extension: Adding Day Names**

Now that you’ve finished the callbacks portion, try extending the WeatherBug graphics! For example, try adding day names to the screen when the weather is displayed. The weather data that you get back has a dayName property that is the day that weather is for. Try displaying another label for each day that displays the name of the day.

****