Task-by-Task Guide

If you'd like a little more support while completing this project, explore this step-by-step resource to get additional hints and resources to help you along each task of this project.

Task 0 - Start with a Plan in Mind

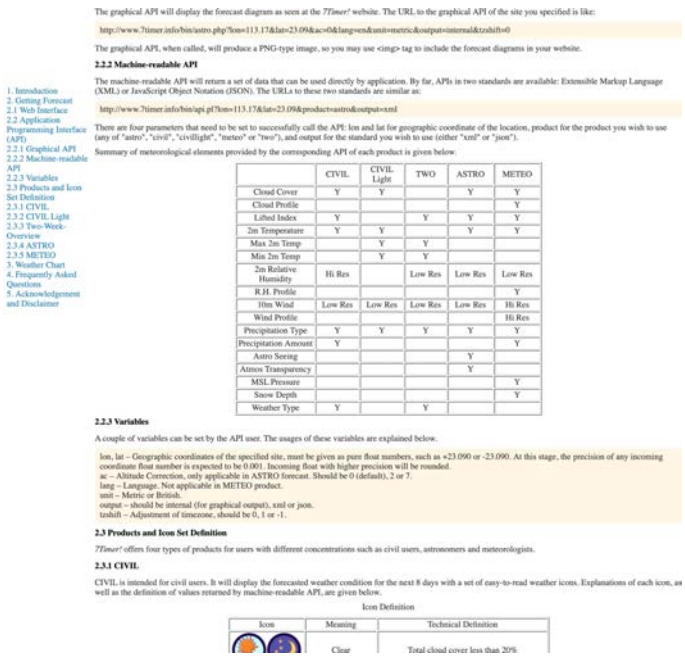
Start With A Plan in Mind

Before you begin, consider taking a step back to plan your steps. Properly planning your project, or scoping, will greatly benefit you; scoping creates structure while requiring you to think through your entire project before you begin. You should start by stating the goals for your project, then gathering the data, and considering the analytical steps required. A proper project scope can be a great road map for your project, but keep in mind that some analyses you start may become dead ends which will require you to adjust your plan.

Task 1 - Explore the Weather Forecast API

The most important step is to explore the API service. What type of parameters do you want to pass along? What is the format of the results will you get back?

In this project, you will be using the 7Timer! API. It is free and does not require any API keys to work. Follow the link and see the different levels of services provided. You will probably be more interested in the Civil types.



1. Introduction
2. Getting Forecast
2.1 Web Interface
2.2 Application Programming Interface (API)
2.2.1 Graphical API
2.2.2 Machine-readable API
2.2.3 Variables
2.3 Products and Icon Set Definition
2.3.1 CIVIL
2.3.2 CIVIL Light
2.3.3 Two-Week-Overview
2.3.4 ASTRO
2.3.5 METEO
3. Weather Chart
4. Frequently Asked Questions
5. Acknowledgement and Disclaimer

The graphical API will display the forecast diagram as seen at the *7Timer!* website. The URL to the graphical API of the site you specified is like:

http://www.7timer.info/bin/astro.php?lon=113.17&lat=23.09&ac=0&lang=en&unit=metric&output=internal&tzshift=0

The graphical API, when called, will produce a PNG-type image, so you may use <img> tag to include the forecast diagrams in your website.

2.2.2 Machine-readable API

The machine-readable API will return a set of data that can be used directly by application. By far, APIs in two standards are available: Extensible Markup Language (XML) or JavaScript Object Notation (JSON). The URLs to these two standards are similar as:

http://www.7timer.info/bin/api.pl?lon=113.17&lat=23.09&product=astro&output=xml

There are four parameters that need to be set to successfully call the API: lon and lat for geographic coordinate of the location, product for the product you wish to use (any of "astro", "civil", "civillight", "meteo" or "two"), and output for the standard you wish to use (either "xml" or "json").

Summary of meteorological elements provided by the corresponding API of each product is given below:

	CIVIL	CIVIL Light	TWO	ASTRO	METEO
Cloud Cover	Y	Y		Y	Y
Cloud Profile					Y
Lifted Index	Y		Y	Y	Y
2m Temperature	Y	Y		Y	Y
Max 2m Temp		Y	Y		
Min 2m Temp		Y	Y		
2m Relative Humidity	Hi Res		Low Res	Low Res	Low Res
R.H. Profile					Y
10m Wind	Low Res	Low Res	Low Res	Low Res	Hi Res
Wind Profile					Hi Res
Precipitation Type	Y	Y	Y	Y	Y
Precipitation Amount	Y				Y
Astro Seeing				Y	
Atmos Transparency				Y	
MSL Pressure					Y
Snow Depth					Y
Weather Type	Y		Y		

2.2.3 Variables

A couple of variables can be set by the API user. The usages of these variables are explained below.

lon, lat – Geographic coordinates of the specified site, must be given as pure float numbers, such as +23.090 or -23.090. At this stage, the precision of any incoming coordinate float number is expected to be 0.001. Incoming float with higher precision will be rounded.
ac – Altitude Correction, only applicable in ASTRO forecast. Should be 0 (default), 2 or 7.
lang – Language. Not applicable in METEO product.
unit – Metric or British.
output – should be internal (for graphical output), xml or json.
tzshift – Adjustment of timezone, should be 0, 1 or -1.

2.3 Products and Icon Set Definition

7Timer! offers four types of products for users with different concentrations such as civil users, astronomers and meteorologists.

2.3.1 CIVIL

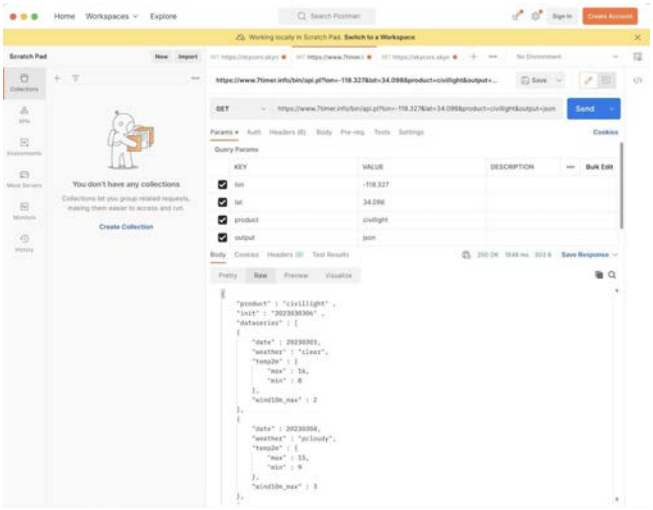
CIVIL is intended for civil users. It will display the forecasted weather condition for the next 8 days with a set of easy-to-read weather icons. Explanations of each icon, as well as the definition of values returned by machine-readable API, are given below.

Icon Definition

Icon	Meaning	Technical Definition
	Clear	Total cloud cover less than 20%

You can use either a web browser (or optionally use an application like Postman) to send requests and examine the results.


```
{
        "product" : "civillight" ,
        "init" : "2023030306" ,
        "dataseries" : [
        {
                "date" : 20230303,
                "weather" : "clear",
                "temp2m" : {
                        "max" : 16,
                        "min" : 8
                },
                "wind10m_max" : 2
        },      {
                "date" : 20230304,
                "weather" : "pcloudy",
                "temp2m" : {
                        "max" : 15,
                        "min" : 9
                },
                "wind10m_max" : 3
        },      {
                "date" : 20230305,
                "weather" : "cloudy",
                "temp2m" : {
                        "max" : 13,
                        "min" : 8
                },
                "wind10m_max" : 3
        },      {
                "date" : 20230306,
                "weather" : "cloudy",
                "temp2m" : {
                        "max" : 13,
                        "min" : 8
```

Task 2 - Apply HTML and CSS to the User Interface

Decide what input you need to gather from users to send to the external service. Do you want to use a dropdown list or find some other way to pass the latitude and longitude of a city along? (A CSV file with latitudes and longitudes of major European cities is included in the starter files).

Design the web page and user interface to accept user input and display results that you get back. If you are well versed in CSS frameworks such as Bootstrap. You may opt to use it.

Also, one of the important concepts of web design is to make your page “responsive”, meaning adapting to different device screen sizes so keep that in mind.

Hints

- If you are using Visual Studio Code, the Live Server extension allows you to see the effects of your code in real time.
- Most modern web browsers (including Google Chrome) have console displays that you can use to debug CSS and JavaScript.
- Be sure to use images that you own or are royalty-free. Using graphics of unknown origin can lead to legal actions and undermine your credibility as a designer or developer. There are a lot of websites where you can find these. For example: Pixabay. You can also search for what you need by adding “royalty free” after the subject in any search engine. Some sites require attribution to use their images. Make sure you follow the instructions.

More Resources:

- Royalty-free graphics: Pixabay
- Free icons (requires attribution): Flat Icons
- Build Fast, Responsive sites with Bootstrap
- Debugging with the Chrome Browser

Task 3 - Write Code to Create and Send API Requests

Write the JavaScript code that builds an HTTP request and sends it to the 7Time! Server. It is recommended that you request data in JSON format since it is easier to work with JavaScript than XML.

If you are well versed in other JavaScript frameworks such as JQuery. You may opt to use it instead of plain JavaScript. Focus on connecting to the external server and getting results back in this step. Don't worry about formatting the results. Simply print it in the browser console or display the raw data on your page.

Hint

- Most modern web browsers (including Google Chrome) have console displays that you can use to debug CSS and JavaScript.
- Remember that HTTP API is asynchronous, meaning you will not be getting the results back right away.

Resources

- Build Fast, Responsive sites with Bootstrap
- Debugging with the Chrome Browser

Task 4 - Process Data and Generate HTML Results

Now it is time to generate the HTML and CSS code that you need to present the results in a way that the users can read. Write the necessary JavaScript to decode the data (JSON or XML) that you get back. Don’t forget to handle any possible errors too.

Hint

Most modern web browsers (including Google Chrome) have console displays that you can use to debug CSS and JavaScript.

More Resources:

- Build Fast, Responsive sites with Bootstrap
- Debugging with the Chrome Browser

Task 6 - Use Fine-Tuning to Test Functionality

Thoroughly test your work and fine-tune any formatting or functionality issues. Make sure all external links work and all your JavaScript work as intended and free of errors. If possible, having someone else test your work might reveal bugs that you won't find otherwise.

Finally, we can wrap up the project. You can write a conclusion about your process and any key findings.

Hint

The main components that you will want to include:

- What did you learn throughout the process?
- Are the results what you expected?
- What are the key findings and takeaways?