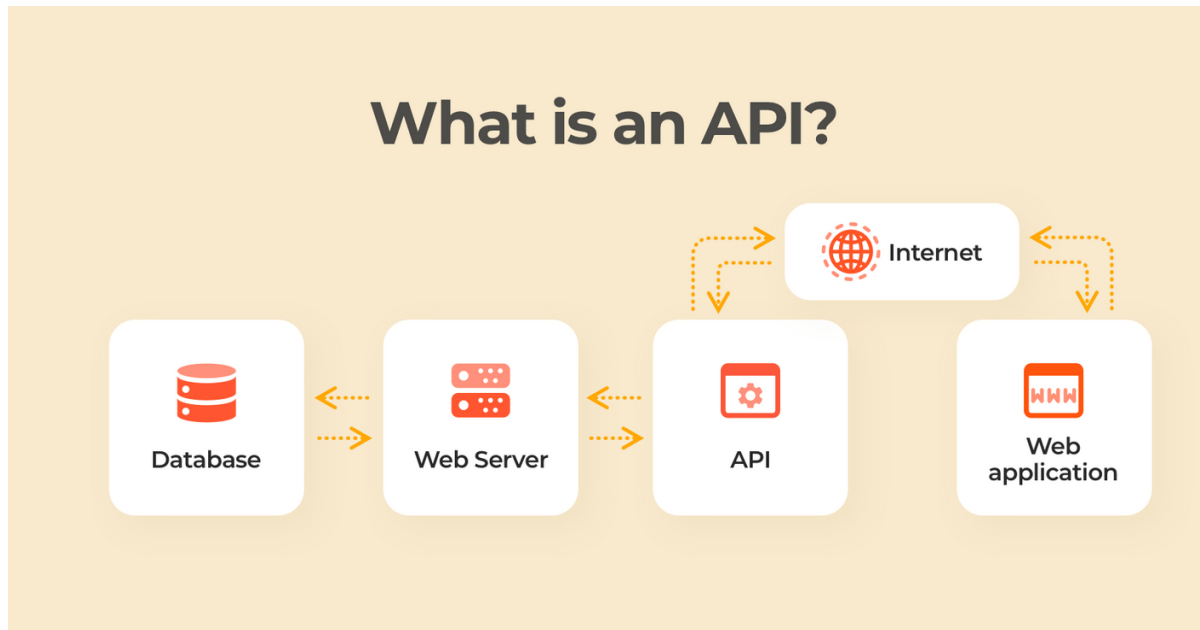


Introduction to API's

Application Programming Interface

By Eli Moser



November 2021 (Ver. 1.000)



Chapter 1: Introduction	3
Chapter 2: Implementation	4
2.1 Example	4



Chapter 1: Introduction

In recent years the concept of API's is gaining popularity, you may have not noticed but they are literally everywhere.

Have you ever wondered how Facebook is able to display your Instagram photos? How about the latest tweet from your favorite celebrity appearing in a news article? Or the latest price of a stock or crypto coin appearing in an article? Your friend sends you his live location on a map that appears in your chat app.

How is all that possible?

The answer: Using an 'Application Programming Interface' or **API**,

Basically using an API is a way for companies to provide information to other computers who can now access and display their information and services.

A high percentage of the data on the internet is passed through API's.

It's a way for all the computers in the world to share information.

Let's say you want an app or a website that provides available flights and hotels (truly an original concept...).

How could we get that information?

The first way would be to manually check all the airlines and Hotels websites and post it in our app for each search - obviously this is very time consuming and not a good option when the app becomes popular (not scalable).

Here comes the API and allows us to connect to all the service providers we want, all we need is to send a request to an API and it will return the information in a way we could automatically display in our app. Fantastic!

Many companies provide API's so that customers could access their information.

Google, Facebook, Tweeter, Yahoo. Finance, weather.com and many other service providers have API's and that's how your Instagram photos could appear on Facebook.

Your weather app on your phone is constantly being updated through an API as well. Cool stuff, now let's see how it's done in the real world.



Chapter 2: Implementation

API's works with Requests and Responses

Request - consists of:

- URL (http://...) – The server address where the database is located,
- Method (GET, POST, PUT, DELETE).
 - GET - used to get a resource from a server.
 - POST - used to create a new resource on a server.
 - PUT - used to update a resource on a server.
 - DELETE - used to delete a resource from a server.
- List of headers - used to provide information to both the client and server. It can be used for many purposes, such as authentication and providing information about the body content.
- Body - used to provide additional information such as media or special parameters

Response – the response from the server in a JSON format.

2.1 Example

We will send a GET request to Skyscanner's API to access available flights:

<https://partners.api.skyscanner.net/apiservices/uk/us/anytime/anytime?apikey=prtl67493879>

We could see here the first part is the URL address:

<https://partners.api.skyscanner.net/apiservices/>

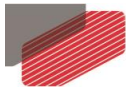
By default the method is set to GET.

The list of headers: the details of the flight locations and dates we are looking for, in our case the locations are UK-US from anytime to anytime:

[/uk/us/anytime/anytime?](#)

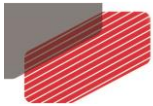
There is also an API key for authentication – it could be seen in the URL

The result is a long list of flights in a JSON format:



```
▼ <BrowseQuotesResponseApiDto>
  ▼ <Quotes>
    ▼ <QuoteDto>
      <QuoteId>1</QuoteId>
      <MinPrice>270</MinPrice>
      <Direct>false</Direct>
      ▼ <OutboundLeg>
        <CarrierIds>1090</CarrierIds>
        <OriginId>82398</OriginId>
        <DestinationId>73076</DestinationId>
        <DepartureDate>2021-11-23T00:00:00</DepartureDate>
      </OutboundLeg>
      ▼ <InboundLeg>
        <CarrierIds>1760</CarrierIds>
        <OriginId>73076</OriginId>
        <DestinationId>82398</DestinationId>
        <DepartureDate>2021-11-28T00:00:00</DepartureDate>
      </InboundLeg>
      <QuoteDateTime>2021-11-08T10:13:00</QuoteDateTime>
    </QuoteDto>
    ▼ <QuoteDto>
      <QuoteId>2</QuoteId>
      <MinPrice>296</MinPrice>
      <Direct>false</Direct>
      ▼ <OutboundLeg>
        <CarrierIds>1760</CarrierIds>
        <OriginId>65655</OriginId>
        <DestinationId>50290</DestinationId>
        <DepartureDate>2022-10-01T00:00:00</DepartureDate>
      </OutboundLeg>
      ▼ <InboundLeg>
        <CarrierIds>1760</CarrierIds>
        <OriginId>50290</OriginId>
        <DestinationId>65655</DestinationId>
        <DepartureDate>2022-10-23T00:00:00</DepartureDate>
      </InboundLeg>
      <QuoteDateTime>2021-11-07T09:39:00</QuoteDateTime>
    </QuoteDto>
    ▼ <QuoteDto>
      <QuoteId>3</QuoteId>
      <MinPrice>315</MinPrice>
      <Direct>true</Direct>
      ▼ <OutboundLeg>
        <CarrierIds>870</CarrierIds>
        <OriginId>65655</OriginId>
        <DestinationId>60987</DestinationId>
        <DepartureDate>2022-01-18T00:00:00</DepartureDate>
      </OutboundLeg>
      ► <InboundLeg>
        ...
      </InboundLeg>
      <QuoteDateTime>2021-11-07T22:35:00</QuoteDateTime>
    </QuoteDto>
    ► <QuoteDto>
      ...
    </QuoteDto>
    ► <QuoteDto>
```

Press on the [Link](#) to see more.



Chapter 3: *scrollTo()* Function

The scrollTo function scrolls the page to a specific location on the page.

A use case would be in a long webpage with multiple sections with links to different sections in the page, instead of reloading the page to the desired section we could use the scrollTo() function which will scroll to the section.

Additional option: scrollAnimation - Indicates whether to scroll with an animation or not.

The WIX's [API](#) documentation describes all the functions in detail.

The first section will describe the function:

scrollTo()

Scrolls the page to a specific location on the page.

Description

The `scrollTo()` function returns a Promise that resolves when the current page has been scrolled to a given location on the page.

The `x` and `y` parameters determine the top-left pixel that is displayed on screen after the scroll.

Tip: To get the coordinates for scrolling, display the Wix Editor Toolbar. In the Editor, move the cursor to the top-left pixel where you want the page to scroll to. The **X** and **Y** axis **Position** values show the coordinates.

To scroll to a specific element on the page, see the `$w.Node.scrollTo()` function.

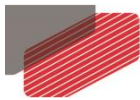
Use the `options` parameter to specify the options to use when scrolling.

Next section will show the way the function is structured:


Syntax

```
function scrollTo(x: number, y: number, [options: ScrollToOptions]): Promise<void>
```

The next section will explain the available parameters:



scrollTo Parameters

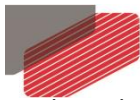
NAME	TYPE	DESCRIPTION
x	number	The horizontal position, in pixels, to scroll to.
y	number	The vertical position, in pixels, to scroll to.
 options OPTIONAL	ScrollToOptions	Scrolling options.
scrollAnimation	boolean	Indicates whether to scroll with an animation. Defaults to <code>true</code> .

Lastly, we have the returned values:

Returns

Fulfilled - When the scroll is complete.

Return Type: Promise<void>



To the right of the screen, you could find some code examples implementing the function:

Scroll the page to a location

[Copy Code](#)

```
1 import wixWindow from 'wix-window';
2
3 // ...
4
5 wixWindow.scrollTo(100, 500);
```

Scroll the page to a location and log message when done

[Copy Code](#)

```
1 import wixWindow from 'wix-window';
2
3 // ...
4
5 wixWindow.scrollTo(100, 500)
6   .then( ( ) => {
7     console.log("Done with scroll");
8   } );
```

Scroll the page to a location without an animation

[Copy Code](#)

```
1 import wixWindow from 'wix-window';
2
3 // ...
4
5 wixWindow.scrollTo(100, 500, {"scrollAnimation": false});
```