

A Comparison of SOAP, REST & GraphQL

by Finley Shaun Clarke

Submission date: 12-Dec-2021 07:50PM (UTC+0000)

Submission ID: 165958586

File name:

32984_Finley_Shaun_Michael_Clarke_A_Comparison_of_SOAP_REST_GraphQL_382340_1382401857.docx
(44.81K)

Word count: 1573

Character count: 8298

Cloud-Based Web Services

A comparison of SOAP, REST and GraphQL

A document detailing the differences and similarities of selected APIs as well as their benefits and disadvantages.

Finley Shaun Michael Clarke (1910794)
Applied Computing YR3 | 2021

Table of Contents

An Introduction to the APIs and What They Are 2

The Comparison..... 4

Referential Information 8

An Introduction to the APIs and What They Are

What is an API?

API stands for Application Programming Interface. It acts as a liaison of sorts between two applications that allow them to communicate with each other and send information and data to each other. They are versatile utilities and can be used on various platforms like web-based systems, operating systems, databases, and even computer hardware.ⁱ They can also be used to access server software or other applications. Some specific examples include payment processing through services like PayPal, universal logins on services such as Facebook, Twitter, YouTube, Google services, and e-commerce websites like Amazon and eBay, as well as services like Google Maps that display static and/or interactive maps.ⁱⁱ

APIs can be dated back to at least the 1950s but modern APIs that are used today can be traced to the early 21st century, more specifically to 2000-2002 where e-commerce sites were beginning to become more and more popular. Their use only continued to increase with the rise of social media and more websites that required accounts.

SOAP

SOAP stands for Simple Object Access Protocol which was designed internally by employees at Microsoft in the late 1990s. It was designed as a standard protocol so that applications designed with differing languages on different platforms were able to communicate and send data to each other. As its status as a protocol, it can impose build-in rules that increases its complexity and overhead. This can result in longer page load times as a result. SOAP can handle requests of data through several means, including HTTP (web browsers), SMTP (for emailing), TCP, as well as some others.ⁱⁱⁱ

GraphQL

The main purpose of GraphQL is as a query language and serve-side runtime for APIs. Introduced by Facebook as an open-source API in 2015 after starting development in 2012, GraphQL was developed as an alternative to REST that favoured mobile development, something that REST was becoming more and more inflexible towards. GraphQL prioritises giving out exclusively the data that clients request from it and absolutely nothing more. It was designed to make APIs faster, more flexible, and consequently more developer friendly. GraphQL is also applicable in integrated development environments (IDEs), notably GraphQL. It is known as a reliable alternative to REST, GraphQL allows developers to make requests that gather data in a single API call, adding to its efficiency.^{iv}



REST

The REST API (sometime referred to as the RESTful API) conforms to the design principles of REST so an architectural style of sorts. REST itself is an architectural style for providing

standards between computer systems on the web, this makes it easier for individual systems to communicate with each other.^v REST API uses HTTP requests to access and use data. This data can then be used to either PUT, GET, POST or DELETE data types. REST tends to use less bandwidth than its peers which makes it more suitable for fluid internet usage. The API is also able to be created using languages like JavaScript and Python.

The Comparison

SOAP Against REST

While technically they achieve the same outcome, their methods are different in several key aspects, SOAP has been relatively phased out by the more current offering of REST. Most websites don't support both protocols as REST provides better services compared to the more restrictive SOAP.

Despite this, SOAP does provide some benefits over REST. For instance, SOAP provides built-in error handling which gives users clear feedback on where errors have occurred once data requests have been sent. REST doesn't have this level of usability and instead just provides simple generic errors instead. This means it's down to the programmer to provide adequate error handling themselves. SOAP also has automation when utilised with certain languages. Also, SOAP is completely independent from any language, platform or transport compared to REST which requires the use of HTTP for example.

REST provides several major advancements over SOAP. The first clear advantage is that REST is simpler to use and easier to grasp compared to SOAP. It is more flexible too. REST doesn't require any expensive tools to interact with a web service. REST also has a much smaller learning curve compared to SOAP; REST is also more efficient than SOAP. Whereas SOAP uses XML for its messaging, REST can make use of smaller messaging formats which allows for more efficient communication as a result. For this same reason, despite there being others too, REST is considered much faster than SOAP too. Finally, REST is more in line with other web technologies in terms of design philosophy which means it more than likely can play nice with them too.

The bottom line for both APIs is that the better of the two is going to depend on what the programmer requires from their chosen API and how much control and flexibility they wish for said API to have. In a lot of cases, the API chosen ends up being REST due to its advancements over SOAP as well as its ease of use and user-friendliness. SOAP isn't without its benefits (chief among them being its built-in error handling which REST does not provide) but REST tends to be the favoured of the two.

REST Against GraphQL

Despite stemming from seemingly different reasons and requirements, REST and GraphQL serve relatively similar purposes, they both send data through protocols like HTTP. How both APIs are implemented is where they both begin to differ.

GraphQL allows users to request and receive the exact information that they want without any complications mostly. REST is less predictable in that manner, as it relies on the HTTP and URL methods that have been put in place. This makes it unclear for the end user on what they're going to receive from the data request. GraphQL is also considered much more simplistic than REST. Despite this, REST is considered more secure and safer to use than GraphQL. Despite having no error-checking capabilities, REST has several ways of

maintaining solid security like enforcing authentications methods like HTTP authentication for example. GraphQL also provides measures although they are not considered as mature as REST. GraphQL does assist programmers in implementing data validation but does not assist in any further authentication and authorisation measures on top, leaving programmers on their own to figure that out. Despite REST still being king, GraphQL has become more and more popular since it's release to the public in 2015. A popularity poll in 2020 showed that despite only 18% of developers/API practitioners using GraphQL, it has been on a steady upwards trajectory since its inception.

Some more differences between REST and GraphQL include: GraphQL utilises a client-driven architecture where REST is solely server-driven. GraphQL only supports JSON as a way of representing data, but REST is able to support various data formats to represent the data it holds. Another key difference is that GraphQL lacks an automatic caching system, but REST is able to do this automatically while handling its processes. GraphQL lacks API versioning while REST supports this right out of the box. Error identification in GraphQL is fairly complex and involves handling HTTP status codes. REST is able to handle and return HTTP status codes, but this functionality is down to the programmer to implement. It should be noted that programmers are able to implement their own error codes as well if they so wish but again this is something down to the programmer and something that REST can't exactly assist in apart from actually outputting said error code when something goes wrong.

As it seems above, GraphQL is on its way to being able to rival REST in terms of usability and functionality but isn't quite there yet and still needs some more years of development. Despite this, more and more programmers and API practitioners are making the jump to take advantage of GraphQL and it's flexibility in mobile development and it's predictability in data returning.

Code Samples

Index.html

```
<!doctype html>
<html lang="en">
  <head>
    <title>
      Exchanging Data using REST and AJAX <!-- Header title -->
    </title>

  </head>
  <body>

    <div class="container">
      <h1>User Data</h1> <!-- Page heading tag -->
      <ul id="list"></ul>
    </div>

    <button type = "button" id="fetchBtn"
      class="btn btn-primary"> <!-- Fetch button for data -->
      Fetch
    </button>
  <script src = "../js/script.js"></script>

  </body>
</html>
```

Script.js

```
//Code below initiates a new object
xhr.open("GET" ,
url, true); //Points towards dummy data

xhr.onload = function ()
{
  if (this.status === 200) // Prepares response
  {
    //Turns string data into JSON object
    obj = JSON.parse(this.responseText) ;

    //Retrieving UL element
    let list = document.getElementById("list") ;
    str = " "
    for (key in obj)
    {
      str += `<li>
      ${obj[key].name} ,
```

```

        ${obj[key].email} <button onClick="showDetail(this)"
id="userBtn" data-id="
        ${obj[key].id}">Get User</button></li>`;
    }
    list.innerHTML = str;
}
}
xhr.send(); //Sends data request
}

function showDetail(obj){
    const xhr = new XMLHttpRequest (); //New XHR object (interacts with
server)

    //Code below initiates a new object
    xhr.open("GET" ,
    url + "/" + obj.getAttribute("data-id"), true); //Points towards dummy
data

    xhr.onload = function ()
    {
        if (this.status === 200) // Prepares response
        {
            //Turns string data into JSON object
            obj = JSON.parse(this.responseText);
            addr = obj.address;

        }
    }
    xhr.send(); //Sends data request
}

```

Referential Information

ⁱ Davis, T., 2019. *What is An API and How Does It Work?* [online] towardsdatascience.com. Available at: <https://towardsdatascience.com/what-is-an-api-and-how-does-it-work-1dccd7a8219e> [Accessed 21 November 2021].

ⁱⁱ "What is an Application Programming Interface (API)", ibm.com, 2020. [Online] Available: <https://www.ibm.com/cloud/learn/api>. [Accessed: 21 Nov 2021].

ⁱⁱⁱ redhat.com. 2019. *REST vs. SOAP*. [online] Available at: <https://www.redhat.com/en/topics/integration/whats-the-difference-between-soap-rest#soap> [Accessed 8 December 2021].

^{iv} redhat.com. 2019. *What is GraphQL?* [online] Available at: <https://www.redhat.com/en/topics/api/what-is-graphql> [Accessed 7 December 2021].

^v codecademy.com. n.d. *What is REST?* [online] Available at: <https://www.codecademy.com/articles/what-is-rest> [Accessed 8 December 2021].

A Comparison of SOAP, REST & GraphQL

GRADEMARK REPORT

FINAL GRADE

48 / 100

GENERAL COMMENTS

Instructor

Ajax & REST (45): 7

Written Research (45): 41

Demo (10): 0

Feedback:
=====

Quite liked the initial discussion of API history. It would have been nice if you had started by identifying some characteristics for comparison.

Some small code samples provided in the word document.

PAGE 1

PAGE 2

PAGE 3



Comment 1

I would like this to be clarified

PAGE 4

PAGE 5

PAGE 6



Comment 2

I would like to see more explanation for this point

PAGE 7

