Web API

JavaScript Console

ECMAScript

Scripting Language

web API

is an application programming interface for either a web server or a web browser. It is a web development concept, usually limited to a web application's client-side (including any web frameworks being used), and thus usually does not include web server or browser implementation details such as SAPIs or APIs unless publicly accessible by a remote web application.

Server side

A server-side web API is a programmatic interface consisting of one or more publicly exposed endpoints to a defined request–response message system, typically expressed in JSON or XML, which is exposed via the web—most commonly by means of an HTTP-based web server. Mashups are web applications which combine the use of multiple server-side web APIs. Webhooks are server-side web APIs that take input as a Uniform Resource Identifier (URI) that is designed to be used like a remote named pipe or a type of callback such that the server acts as a client to dereference the provided URI and trigger an event on another server which handles this event thus providing a type of peer-to-peer IPC.

Endpoints

Endpoints are important aspects of interacting with server-side web APIs, as they specify where resources lie that can be accessed by third party software. Usually the access is via a URI to which HTTP requests are posted, and from which the response is thus expected. Web APIs may be public or private, the latter of which requires an *access token*.

Endpoints need to be static, otherwise the correct functioning of software that interacts with it cannot be guaranteed. If the location of a resource changes (and with it the endpoint) then previously written software will break, as the required resource can no longer be found at the same place. As API providers still want to update their web APIs, many have introduced a versioning system in the URI that points to an endpoint, for example the Clarifai API: The endpoint for the tagging functionality within the web API has the following URI: "https://api.google.com/v1/tag/". The "/v1/" part of the URI specifies access to the first version of the web API. If Clarifai decides to update to version two, they can do this while still maintaining support for third party software that uses the first version.

Resources versus services

Web 2.0 Web APIs often use machine-based interactions such as REST and SOAP. RESTful web APIs are typically loosely based on HTTP methods to access resources via URL-encoded parameters and the use of JSON or XML to transmit data. By contrast, SOAP protocols are standardized by the W3C and mandate the use of XML as the payload format, typically over HTTP. Furthermore, SOAP-based Web APIs use XML validation to ensure structural message integrity, by leveraging the XML schemas provisioned with WSDL documents.
A WSDL document accurately defines the XML messages and transport bindings of a Web service.

Documentation

Server-side web APIs are interfaces for the outside world to interact with the business logic. For many companies this internal business logic and the intellectual property associated with it are what distinguishes them from other companies, and potentially what gives them a competitive edge. They do not want this information to be exposed. However, in order to provide a web API of high quality, there *needs* to be a sufficient level of documentation. One API provider that not only provides documentation, but also links to it in its error messages is Twilio.

However, there are now directories of popular documented server-side web APIs.

Growth and impact

The number of available web APIs has grown consistently over the past years, as businesses realize the growth opportunities associated with running an open platform, that any developer can interact with. ProgrammableWeb tracks 9000 Web APIs that were available in 2013, up from 105 in 2005.

Web APIs have become ubiquitous. There are few major software applications/services that do not offer some form of web API. One of the most common forms of interacting with these web APIs is via embedding external resources, such as tweets, Facebook comments, YouTube videos, etc. In fact there are very successful companies, such as Disqus, whose main service is to provide embeddable tools, such as a feature-rich comment system. Any website of the TOP 100 Alexa Internet ranked websites uses APIs and/or provides its own APIs, which is a very distinct indicator for the prodigious scale and impact of web APIs as a whole.

As the number of available web APIs has grown, open source tools have been developed to provide more sophisticated search and discovery. APIs.json provides a machine-readable description

of an API and its operations, and the related project APIs.io offers a searchable public listing of APIs based on the APIs.json metadata format.

Commercial

Many companies and organizations rely heavily on their Web API infrastructure to serve their core business clients. In 2014 Netflix received around 5 billion API requests, most of them within their private API.

Client side

A client-side web API is a programmatic interface to extend functionality within a web browser or other HTTP client. Originally these were most commonly in the form of native plug-in browser extensions however most newer ones target standardized JavaScript bindings.

The Mozilla Foundation created their WebAPI specification which is designed to help replace native mobile applications with HTML5 applications.

Google created their Native Client architecture which is designed to help replace insecure native plug-ins with secure native sandboxed extensions and applications. They have also made this portable by employing a modified LLVM AOT compiler.

console

The console object provides access to the browser's debugging console (e.g. the Web console in Firefox). The specifics of how it works varies from browser to browser, but there is a de facto set of features that are typically provided.

The console object can be accessed from any global object. Window on browsing scopes and WorkerGlobalScope as specific variants in workers via the property console. It's exposed as Window.console, and can be referenced as console. For example:

```
console.log("Failed to open the specified link")
```

Methods

console.assert()

Log a message and stack trace to console if the first argument is false.

console.clear()

Clear the console.

console.count()

Log the number of times this line has been called with the given label.

console.countReset()

Resets the value of the counter with the given label.

console.debug()

Outputs a message to the console with the log level debug.

console.dir()

Displays an interactive listing of the properties of a specified JavaScript object. This listing lets you use disclosure triangles to examine the contents of child objects.

console.dirxml()

Displays an XML/HTML Element representation of the specified object if possible or the JavaScript Object view if it is not possible.

console.error()

Outputs an error message. You may use string substitution and additional arguments with this method.

console.exception()

An alias for error().

console.group()

Creates a new inline group, indenting all following output by another level. To move back out a level, call groupEnd().

console.groupCollapsed()

Creates a new inline group, indenting all following output by another level. However, unlike group() this starts with the inline group collapsed requiring the use of a disclosure button to expand it. To move back out a level, call groupEnd().

console.groupEnd()

Exits the current inline group.

console.info()

Informative logging of information. You may use string substitution and additional arguments with this method.

console.log()

For general output of logging information. You may use string substitution and additional arguments with this method.

console.profile()

Starts the browser's built-in profiler (for example, the Firefox performance tool). You can specify an optional name for the profile.

console.profileEnd()

Stops the profiler. You can see the resulting profile in the browser's performance tool (for example, the Firefox performance tool).

console.table()

Displays tabular data as a table.

console.time()

Starts a timer with a name specified as an input parameter. Up to 10,000 simultaneous timers can run on a given page.

console.timeEnd()

Stops the specified timer and logs the elapsed time in milliseconds since it started.

console.timeLog()

Logs the value of the specified timer to the console.

console.timeStamp()

Adds a marker to the browser's Timeline or Waterfall tool.

console.trace()

Outputs a stack trace.

console.warn()

Outputs a warning message. You may use string substitution and additional arguments with this method.

ECMAScript

JavaScript language resources

ECMAScript is the scripting language that forms the basis of JavaScript. ECMAScript standardized by the ECMA International standards organization in the ECMA-262 and ECMA-402 specifications. The following ECMAScript standards have been approved or are being worked on:

Name	Links	Release date	Description
Current editions			
ECMA-262 11th Edition	PDF, HTML, Working draft, repository	2020	ECMAScript 2020 Language Specification
ECMA-262 10th Edition	PDF, HTML, Working draft, repository	2019	ECMAScript 2019 Language Specification
ECMA-262 9th Edition	PDF, HTML, Working draft, repository	2018	ECMAScript 2018 Language Specification
ECMA-402 5th Edition	Working draft, repository	2018	ECMAScript 2018 Internationalization API Specification
Obsolete/historical editions			
ECMA-262	PDF	June 1997	ECMAScript: A general purpose, cross-platform programming language. This was the first version of the ECMAScript standard.
ECMA-262 2nd Edition	PDF	August 1998	ECMAScript Language Specification. This is the second revision of the ECMAScript standard; also ISO standard 16262.
ECMA-262 3rd Edition	PDF	December 1999	ECMAScript Language Specification. This is the third revision of the ECMAScript standard; corresponds to JavaScript 1.5. See also the errata

ECMA-262 5th Edition	PDF	December 2009	ECMAScript Language Specification. This is the fifth revision of the ECMAScript standard. See also the ES5 errata and ECMAScript 5 support in Mozilla
ECMA-262 5.1 Edition	PDF, HTML	June 2011	This version is fully aligned with the third edition of the international standard ISO/IEC 16262:2011. It includes ES5 errata fixes, no new features.
ECMA-402 1st Edition	PDF, HTML	December 2012	ECMAScript Internationalization API Specification
ECMA-262 6th Edition	PDF, HTML	June 2015	ECMAScript 2015 Language Specification
ECMA-402 2nd Edition	PDF	June 2015	ECMAScript 2015 Internationalization API Specification
ECMA-262 7th Edition	HTML	June 2016	ECMAScript 2016 Language Specification
ECMA-402 3rd Edition	HTML	June 2016	ECMAScript 2016 Internationalization API Specification
ECMA-262 8th Edition	HTML	June 2017	ECMAScript 2017 Language Specification
ECMA-402 4th Edition	HTML	June 2017	ECMAScript 2017 Internationalization API Specification

---

Scripting Language

What Is a Scripting Language?

A scripting language is a programming language that is interpreted. It is translated into machine code when the code is

run, rather than beforehand. Scripting languages are often used for short scripts over full computer programs. JavaScript, Python, and Ruby are all examples of scripting languages.

Scripting language (also known as scripting, or script) is a series of commands that are able to be executed without the need for compiling. While all scripting languages are programming languages, not all programming languages are scripting languages. PHP, Perl, and Python are common examples of scripting languages.

Scripting languages use a program known as an interpreter to translate commands and are directly interpreted from source code, not requiring a compilation step. Other programming languages, on the other hand, may require a *compiler* to translate commands into machine code before it can execute those commands.

It is important to know the difference between interpreted vs compiled programming languages. However, advanced hardware and coding practices are beginning to make the distinction somewhat obsolete.

Examples of Server-Side Scripting Languages

The following are examples of server-side scripting languages.

Language	Comments
PHP	The most popular server-side language used on the web.
ASP.NET	Web-application framework developed by Microsoft.
Node.js	Can run on a multitude of platforms, including Windows, Linux, Unix, Mac, etc.
Java	Used in everything from your car stereo's Bluetooth to NASA applications.

Ruby	Dynamic. Focuses heavily on simplicity.
Perl	A bit of a mashup between C, shell script, AWK, and sed.
Python	Great for beginners to learn. Uses shorter code.