XML

XML (Extensible Markup Language) is a markup language similar to HTML, but without predefined tags to use. Instead, you define your own tags designed specifically for your needs. This is a powerful way to store data in a format that can be stored, searched, and shared. Most importantly, since the fundamental format of XML is standardized, if you share or transmit XML across systems or platforms, either locally or over the internet, the recipient can still parse the data due to the standardized XML syntax.

There are many languages based on XML, including XHTML, MathML, SVG, RSS, and RDF. You can also define your own.

API

Application Programming Interfaces (APIs) are constructs made available in programming languages to allow developers to create complex functionality more easily. They abstract more complex code away from you, providing some easier syntax to use in its place.

As a real-world example, think about the electricity supply in your house, apartment, or other dwellings. If you want to use an appliance in your house, you plug it into a plug socket and it works. You don't try to wire it directly into the power supply — to do so would be really inefficient and, if you are not an electrician, difficult and dangerous to attempt.

APIs in client-side JavaScript

Client-side JavaScript, in particular, has many APIs available to it — these are not part of the JavaScript language itself, rather they are built on top of the core JavaScript language, providing you with extra superpowers to use in your JavaScript code. They generally fall into two categories:

Browser APIs are built into your web browser and are able to expose data from the browser and surrounding computer environment and do useful complex things with it. For example, the Web Audio API provides JavaScript constructs for manipulating audio in the browser — taking an audio track, altering its volume, applying effects to it, etc. In the background, the browser is actually using some complex lower-level code (e.g. C++ or Rust) to do the actual audio processing. But again, this complexity is abstracted away from you by the API.

Third-party APIs are not built into the browser by default, and you generally have to retrieve their code and information from somewhere on the Web. For example, the Google Maps API allows you to do things like display an interactive map to your office on your website. It provides a special set of constructs you can use to query the Google Maps service and return specific information.

Relationship between JavaScript, APIs, and other JavaScript tools

JavaScript — A high-level scripting language built into browsers that allows you to implement functionality on web pages/apps. Note that JavaScript is also available in other programming environments, such as Node.

Browser APIs — constructs built into the browser that sit on top of the JavaScript language and allow you to implement functionality more easily.

Third-party APIs — constructs built into third-party platforms (e.g. Disqus, Facebook) that allow you to use some of those platform's functionality in your own web pages (for example, display your Disqus comments on a web page).

JavaScript libraries — Usually one or more JavaScript files containing custom functions that you can attach to your web page to speed up or enable writing common functionality. Examples include jQuery, Mootools and React.

JavaScript frameworks — The next step up from libraries, JavaScript frameworks (e.g. Angular and Ember) tend to be packages of HTML, CSS, JavaScript, and other technologies that you install and then use to write an entire web application from scratch. The key difference between a library and a framework is "Inversion of Control". When calling a method from a library, the developer is in control. With a framework, the control is inverted: the framework calls the developer's code.

Common browser APIs

APIs for manipulating documents loaded into the browser. The most obvious example is the DOM (Document Object Model) API, which allows you to manipulate HTML and CSS — creating, removing and changing HTML, dynamically applying new styles to your page, etc. Every time you see a popup window appear on a page or some new content displayed, for example, that's the DOM in action. Find out more about these types of API in Manipulating documents.

APIs that fetch data from the server to update small sections of a webpage on their own are very commonly used. This seemingly small detail has had a huge impact on the performance and behavior of sites — if you just need to update a stock listing or list of available new stories, doing it instantly without having to reload the whole entire page from the server can make the site or app feel much more responsive and "snappy". The main API used for this is the Fetch API, although older code might still use the XMLHttpRequest API. You may also come across the term Ajax, which describes this technique. Find out more about such APIs in Fetching data from the server.

APIs for drawing and manipulating graphics are widely supported in browsers — the most popular ones are Canvas and WebGL, which allow you to programmatically update the pixel data contained in an HTML <canvas> element to create 2D and 3D scenes. For example, you might draw shapes such as rectangles or circles, import an image onto the canvas, and apply a filter to it such as sepia or grayscale using the Canvas API, or create a complex 3D scene with lighting and textures using WebGL. Such APIs are often combined with APIs for creating animation loops (such as window.requestAnimationFrame()) and others to make constantly updating scenes like cartoons and games.

Audio and Video APIs like HTMLMediaElement, the Web Audio API, and WebRTC allow you to do really interesting things with multimedia such as creating custom UI controls for playing audio and video, displaying text tracks like captions and subtitles along with your videos, grabbing video from your web camera to be manipulated via a canvas (see above) or displayed on someone else's computer in a web conference, or adding effects to audio tracks (such as gain, distortion, panning, etc.).

Device APIs enable you to interact with device hardware: for example, accessing the device GPS to find the user's position using the Geolocation API.

Client-side storage APIs enable you to store data on the client-side, so you can create an app that will save its state between page loads, and perhaps even work when the device is offline. There are several options available, e.g. simple name/value storage with the Web Storage API, and more complex database storage with the IndexedDB API.

Common third-party APIs

Map APIs, like Mapquest and the Google Maps API, which allow you to do all sorts of things with maps on your web pages.

The Facebook suite of APIs, which enables you to use various parts of the Facebook ecosystem to benefit your app, such as by providing app login using Facebook login, accepting in-app payments, rolling out targeted ad campaigns, etc.

The Telegram APIs, which allows you to embed content from Telegram channels on your website, in addition to providing support for bots.

The YouTube API, which allows you to embed YouTube videos on your site, search YouTube, build playlists, and more.

The Pinterest API, which provides tools to manage Pinterest boards and pins to include them in your website.

The Twilio API, which provides a framework for building voice and video call functionality into your app, sending SMS/MMS from your apps, and more.

The Disqus API, which provides a commenting platform that can be integrated into your site.

The Mastodon API, which enables you to manipulate features of the Mastodon social network programmatically.

The IFTTT API, which enables integrating multiple APIs through one platform.

XMLHTTPRequest

XMLHttpRequest (XHR) objects are used to interact with servers. You can retrieve data from a URL without having to do a full page refresh. This enables a Web page to update just part of a page without disrupting what the user is doing.

XMLHttpRequest()

The constructor initializes an XMLHttpRequest. It must be called before any other method calls.

Instance properties

This interface also inherits properties of XMLHttpRequestEventTarget and of EventTarget.

XMLHttpRequest.readyState

Returns a number representing the state of the request.

XMLHttpRequest.response

Returns an ArrayBuffer, a Blob, a Document, a JavaScript object, or a string, depending on the value of XMLHttpRequest.responseType, that contains the response entity body.

XMLHttpRequest.responseText

Returns a string that contains the response to the request as text, or null if the request was unsuccessful or has not yet been sent.

XMLHttpRequest.responseType

Specifies the type of the response.

XMLHttpRequest.responseURL

Returns the serialized URL of the response or the empty string if the URL is null.

XMLHttpRequest.responseXML

Returns a Document containing the response to the request, or null if the request was unsuccessful, has not yet been sent, or cannot be parsed as XML or HTML. Not available in Web Workers.

XMLHttpRequest.status

Returns the HTTP response status code of the request.

XMLHttpRequest.statusText

Returns a string containing the response string returned by the HTTP server. Unlike XMLHttpRequest.status, this includes the entire text of the response message ("OK", for example).

XMLHttpRequest.timeout

The time in milliseconds a request can take before automatically being terminated.

XMLHttpRequest.upload

A XMLHttpRequestUpload representing the upload process.

XMLHttpRequest.withCredentials

Returns true if cross-site Access-Control requests should be made using credentials such as cookies or authorization headers; otherwise false.

<u>XMLHttpRequest: readyState property</u>

The **XMLHttpRequest.readyState** property returns the state an XMLHttpRequest client is in. An XHR client exists in one of the following states:

Value	State	Description
0	UNSENT	Client has been created. open() not called yet.
1	OPENED	open() has been called.
2	HEADERS_RECEIVED	send() has been called, and headers and status are available.
3	LOADING	Downloading; responseText holds partial data.
4	DONE	The operation is complete.

UNSENT

The XMLHttpRequest client has been created, but the open() method hasn't been called yet.

OPENED

open() method has been invoked. During this state, the request headers can be set using the setRequestHeader() method and the send() method can be called which will initiate the fetch.

HEADERS_RECEIVED

send() has been called, all redirects (if any) have been followed and the response headers have been received.

LOADING

Response's body is being received. If responseType is "text" or empty string, responseText will have the partial text response as it loads.

DONE

The fetch operation is complete. This could mean that either the data transfer has been completed successfully or failed.