Web Development Notes

Alex Colwell

February 2, 2022

Contents

The Internet's Structure

0.1 Web Servers

Web servers are where we usually get web pages' files. They are also where our sites' databases, APIs, and all our other webs stuff lives. Basically, if we want it to be available over the web, we put it on a server.

0.2 Web Clients

Clients are what we use to visit websites. Some examples are Firefox, Safari, Chrome, and Opera. Their main jobs are to make requests to servers and display the results. They ask for whatever site we're looking for's public files (usually HTML, CSS, and Javascript), and use said files to display web pages.

0.3 APIs

Application Programming Interfaces (APIs) are how we send and receive non-page-related data to and from web servers. For example, there are APIs for getting weather data as JSON.

0.4 NTP servers

Network Time Protocol (NTP) servers are what computers use to synchronize their times. All they do is tell the current time. There are different stratum of NTP servers, with the ones closer to zero being closer to whatever we're using as "true time," for example GPS antennas.

0.5 DNS servers

Domain Name Servers are what we use to convert domains (e.g.www.google.com) to IP addresses.

Client–Server Communication

0.6 Client to Server

Communication initiated by the client is pretty straightforward: the client sends some kind of request (e.g. get, post, etc.) to the server, and the server responds.

0.7 Server to Client

In today's most common HTTP versions (1.1 and 2), servers can't initiate communications with a client. This can be a problem, since sometimes the server needs to send data to the client without the client first sending a request. Luckily, there are a few methods for getting around this.

0.7.1 Sockets

Sockets are the most capable of these methods. It allows either side to send and receive data, is real-time, and widely supported. It basically works like this: both the client and server connect to a socket server, (which can be on the same hardware as the web server or provided by a third party). Then they either listen on or send data to a channel (kinda like an IRC channel).

0.7.2 Long Polling

Long polling is probably the next best solution (assuming you need real-time communication). The client sends a request for the data and the server holds it open, and then responds to it when the data is available. Then the client sends another request for the data, and on and on. This way the client gets the data as soon as the server is ready to send it.

0.7.3 SSE

Server Side Events (SSE) are kinda like sockets, but they're only one way (server to client). The client uses the "EventSource" interface, which listens to some endpoint and emits events when some data is sent. On the server side, when the EventSource object makes a request to whatever endpoint we're using, it returns an event-stream, through which it sends data.

0.7.4 Polling

Polling is the simplest of these options: the client just makes a request for the data every few seconds.