How the Web Functions

This document describes what happens when a user types a URL in their browser and hits enter. It will explain how data reaches you to be rendered in the browser, what code is rendered in the browser, and what are the main functions of server-side code vs. client-side code.

Server-side vs. Client-side code and Runtime

Code running in the browser is client-side code and mostly controls structure and presentation of data to the user. Things like selecting and styling UI components, creating layouts, navigation, form validation, etc. Server-side code involves choosing which content is returned to the browser. This includes validating submitted data and requests, databases queries to store and retrieve data etc. Runtime describes software/instructions that are executed while your program is running, especially those instructions that you did not write explicitly, but are needed for proper execution of your code.

What happens when a user requests a URL in a browser

1. User types the URL (https://www.techtonicgroup.com/) into the address bar of their browser. Every URL on the internet has a unique IP (ex: 71.218.57.94) address assigned to it. DNS (Domain Name System) is a database that maintains a reference between the URL and IP address.

2. The browser checks its cache for a DNS record to find the corresponding IP address of www.techtonicgroup.com/.

   The browser checks four caches:
   - Browser cache: The browser stores DNS records of websites you have previously visited for a fixed duration of time.
   - OS cache: The browser makes a system call to your computer's OS to fetch the DNS record.
   - Router cache: The browser checks with the router, which also maintains a cache of DNS records.
   - ISP cache: Your ISP maintains its own DNS server which includes a cache of DNS records.

3. If the requested URL is not in the cache, the ISP's DNS server initiates a DNS query to find the IP address of the server that hosts www.techtonicgroup.com. The DNS query will search multiple DNS servers until it finds the correct IP address.

4. Browser initiates a TCP connection with the server.
   Once the browser receives the correct IP address, it builds a connection with the server to transfer information. TCP is the most common protocol used for any type of HTTP request and is established using the TCP/IP three-way handshake where the client and the server exchange SYN (synchronize) and ACK (acknowledge) messages to establish a connection:
   - Client machine sends a SYN packet to the server asking if it is open for new connections.
   - If the server has open ports that can accept and initiate new connections, it responds with an ACKnowledgment of the SYN packet using a SYN/ACK packet.
   - The client receives the SYN/ACK packet and acknowledges it by sending an ACK packet.

5. The browser sends an HTTP request to the web server.
   Once the TCP connection is established, the browser will send a GET request asking for the www.techtonicgroup.com web page. This request will also contain additional information such as browser identification (User-Agent header), types of requests that it will accept (Accept header), and connection headers asking it to keep the TCP connection alive for additional requests. It will also pass information taken from cookies the browser has in store for this domain. Sample GET request (Headers are highlighted):

```
GET http://facebook.com/ HTTP/1.1
Accept: application/x-ms-application, image/jpeg, application/xaml+xml, [...]
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; [...]
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
Host: facebook.com
Cookie: datr=1265876274-[...]; locale=en_US; lsd=WW[...]; c_user=2101[...]
```

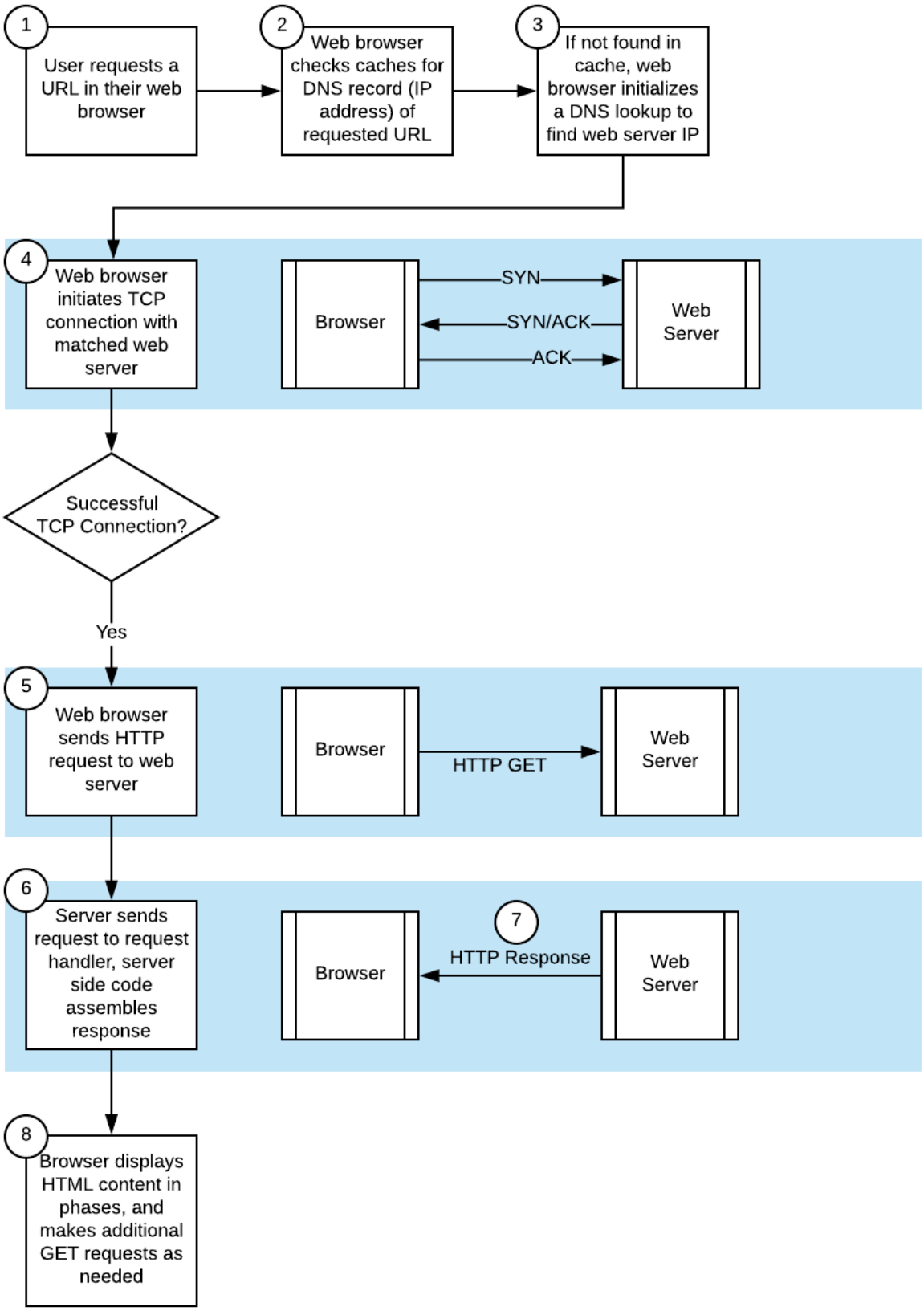
6. The server handles the request and sends back a response.
   The server contains a web server (i.e. Apache, IIS) which receives the request from the browser and passes it to a request handler to read and generate a response. The request handler is a program (written in ASP.NET, PHP, Ruby, etc.) that reads the request, headers, and cookies to check what is being requested and also update the information on the server if needed. Then it will assemble a response in a particular format (JSON, XML, HTML). The web server can handle many instances of server-side code and database connections as needed to handle multiple browser requests.

7. The server sends out an HTTP response.
   The server response contains the web page you requested as well as the status code, compression type (Content-Encoding), how to cache the page (Cache-Control), any cookies to set, privacy information, etc. One instance of each client-side asset needed is sent to the requesting browser.

   Example HTTP server response:

```
HTTP/1.1 200 OK
Cache-Control: private, no-store, no-cache, must-revalidate, post-check=0,
    pre-check=0
Expires: Sat, 01 Jan 2000 00:00:00 GMT
P3P: CP="DSP LAW"
Pragma: no-cache
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
X-Cnection: close
Transfer-Encoding: chunked
Date: Fri, 12 Feb 2010 09:05:55 GMT
```

8. The browser displays the HTML content.
   The browser renders the HTML content in phases. First, it will render the bare bone HTML skeleton. Then it will check the HTML tags and send out GET requests for additional elements on the web page, such as images, CSS stylesheets, JavaScript files etc. These static files are cached by the browser so it doesn't have to fetch them again the next time you visit the page.


1
User requests a URL in their web browser
2
Web browser checks caches for DNS record (IP address) of requested URL
3
If not found in cache, web browser initializes a DNS lookup to find web server IP
4
Web browser initiates TCP connection with matched web server
Browser
SYN
SYN/ACK
ACK
Web Server
Successful TCP Connection?
Yes
5
Web browser sends HTTP request to web server
Browser
HTTP GET
Web Server
6
Server sends request to request handler, server side code assembles response
7
Browser
HTTP Response
Web Server
8
Browser displays HTML content in phases, and makes additional GET requests as needed