What is Web, Client Server Architecture, and HTTP Protocol

Guillermo Ventura-Reyes
Anton Marku

Brief History

- The web started in the 1960s when the US military developed ARPANET for communication purposes. It used packet switching, and the TCP/IP protocol.
- Tim Berners Lee created a program demonstrating linking between different nodes. He would later, in late 1990, create HTTP, HTML, a web browser, an HTTP server, and some web pages to explore.
- The web would go on to explode in popularity, resulting in what we use today, and also the dotcom bubble.
- The World Wide Web Consortium,or W3C was founded by TimBL in 1994 would yield the key web technologies, CSS and JavaScript that are widely used today.

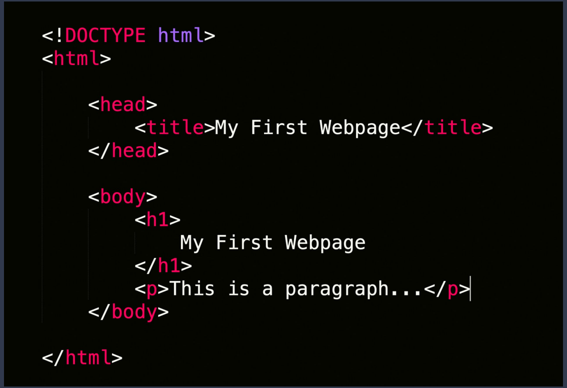
Web Standards

- Web standards are long technical documents known as specifications which detail how technologies utilized to build websites should work. Those documents are most used the the engineers who actually work on web technologies.
- One example is the HTML Living Standard which specifies precisely how HTML and associated APIs should be implemented.
- Web Standards are created by standards institutions which invite experts in the field do discuss the best way to implement these technologies based on their intended purposes.
- WHATWG maintains the HTML Living Standard. ECMA publish the ECMAScript standard, which is the basis of JavaScript. Khronos publish 3d graphics tech, like WebGL.

“Open” Standards and Web Technology Etiquette

- TimBL and W3C agreed that the web as a whole should be free to contribute to and use, without burdensome patents slowing down progress.
- The idea is for the progress, creation, and distribution of the web to be an open, democratic progress rather than allow a handful of companies to dominate and restrict the freedom of all the regular people who want to use the internet. It is meant to be a free public resource.
- Technology etiquette dictates that one not “break” the web. Newer web technologies should still support older websites, and future technologies should work with what is available today.
- Web development is more complicated than it’s ever been, with websites being required to support various things like millions of concurrent users, massive amounts of user generated content(like youtube), various forms of content delivery(text, video, live stream, images), a purchase processing system and more.
- *“The only constant is change.”*

Browsers, HTTP, & HTML/CSS/JavaScript

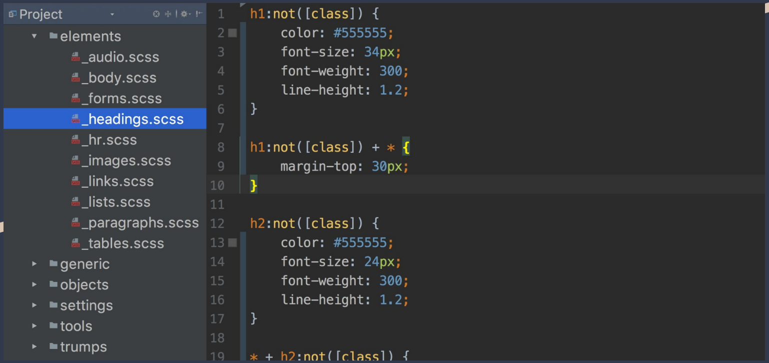


```
<!DOCTYPE html>
<html>

    <head>
        <title>My First Webpage</title>
    </head>

    <body>
        <h1>
            My First Webpage
        </h1>
        <p>This is a paragraph...</p>
    </body>

</html>
```



```
h1:not([class]) {
    color: #555555;
    font-size: 34px;
    font-weight: 300;
    line-height: 1.2;
}

h1:not([class]) + * {
    margin-top: 30px;
}

h2:not([class]) {
    color: #555555;
    font-size: 24px;
    font-weight: 300;
    line-height: 1.2;
}

* + h2:not([class]) {
```

- Browsers are the programs one can use to go to different websites. Firefox, Chrome, Opera, Safari, and Edge are some of the major ones.
- Hypertext Transfer Protocol is a messaging protocol that handles browser-server communication. The server has the website information and all the content available on that website. The browser sends an HTTP request for some information and receives a response.
- The purpose of HTML is to determine the general structure of your website. You can think of this as similar to the blueprint of a house. CSS allows you to finetune that structure, defining spacing between elements, arrangement, font color, size, animations, and more.
- Javascript is the functionality of the website. It can modify website style dynamically, fetch server updates, display 3d graphics, and more. This can be thought of as house appliances.
- These are all client side technologies in that they run on the client's local machine.

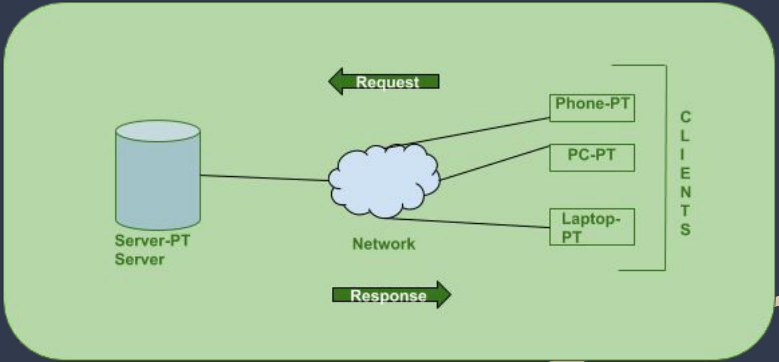
Tooling and Server Side Languages/ Frameworks

- HTML/CSS/JavaScript can be thought of as the raw materials for building a websites. Developers use additional tools that make the process of web development easier.
- Developer tools inside modern browsers can debug code.
- Testing tools can run tests that determine whether your website is behaving properly.
- JavaScript libraries and frameworks can be useful for specific types of websites.
- Linters use a given set of rules to determine where in your code you failed to follow those rules.
- Minifiers remove whitespace from code files to reduce their size in bytes and thus allow them to be downloaded faster.
- Backend languages work in the server and prepare the information that will be sent to the user. This might involve accessing the requested information in a database, preparing some HTML to hold the data, and sending that to the user.
- ASP.NET, Python, PHP, and Node.js are backend languages/frameworks.

Web Best Practices

- Web sites can be accessed by anyone, anywhere, with all kinds of devices, internet connections, and personal circumstances that affect how they will react to a website. All of these must be considered when developing a website.
- Users could be using a mobile device, a PC with a widescreen monitor, they might be blind using a screen reader, color blind, or using a very old computer that does not support modern browsers.
- Developers should aim for compatibility across browsers and versions. The design of their sites should be responsive to different screen types for optimal viewing.
- Websites should be fast but still user friendly to avoid frustrating the user. Accessibility, which includes diversity and inclusion should also be considered. People of various ages, with various disabilities, and from all different countries(Internationalization).
- Users should have privacy when browsing the internet. Any data that they store on the website should also be kept secure and not at risk of being stolen from malicious users.

Client Server Architecture



- A computing model in which the server hosts, delivers, and manages most of the resources to be consumed the client.
- Has one or more clients connected to the central server via a network or internet connection
- **Client:** software/application that takes inputs and sends requests to the server
- **Server:** software the receives and processes requests from client
- The other popular architecture is Peer-To-Peer

Advantages and Disadvantages

Advantages

- Centralized system with all data in a single place
- Cost efficient, requires less maintenance cost and data recovery is possible
- Capacity of client and server can be changed separately

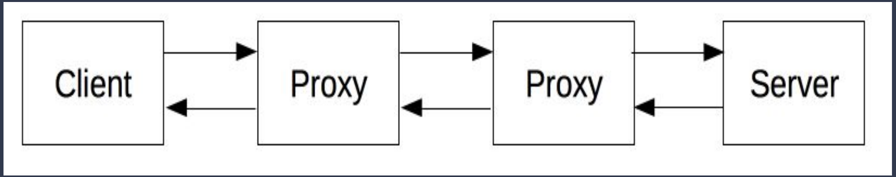
Disadvantages

- Clients are prone to viruses, Trojans, and worms if present or added to the server
- Servers are prone to DDOS attacks
- Phishing /capturing login credentials are common

HyperText Transfer Protocol

- A protocol for the fetching of resources
  - A protocol is a system of rules to define how data is exchanged between computers
  - It is a client server protocol
- Requests are initiated by the recipient (Web Browser)
- Constructs a document from multiple sub-documents (text, layout description, imgs, vids, scripts, etc…)
- Communicate by sending individual messages
  - Messages sent by the client are requests
  - Messages sent by the server as an answer are responses

HTTP Components



- Because HTTP is a client-server protocol, requests are sent by one entity, the user entity (or a proxy). Most commonly the user entity is a web browser but it could be anything, including bots crawling and scraping/indexing the web (like for a search engine)
- Every individual request is sent to the server, which handles the request and provides a response. There may be numerous entities between them named proxies that perform different operation and act as gateways/caches
- In reality there are numerous computers handling the requests between the browser and server such as routers,modems, and thanks to the design of the web HTTP is on top of these layers. The Underlying layers are irrelevant in the description of HTTP

HTTP Components: Client - the user agent

- Any tool that acts on behalf of the user (primarily the browser but other programs exists)
- The browser is always the entity initiating the request
- To present the web page, the browser sends an original request to fetch , it parses the returned HTML document, making additional requests to execution scripts (e.g. Javascript), layout info (CSS), and sub-resources (imgs and vids)
- This web page is a hypertext document, having hyperlinks which can be clicked and fetch a new web page. The browser translates these in HTTP requests and keeps interpreting HTTP responses to the user
    -

HTTP Components: Server

- Serves the document as requested by client.

- Appears as a single machine virtually, but may actually be a collection of servers sharing the load(load balancing) or another piece of software that interacts with other computers (cache, DB server, e-commerce servers) generating the document on demand

- Several servers can also be hosted on the machine, and may even share the same IP address.

HTTP Components: Proxy

- computers/machines that operate on the application layer that relay the HTTP messages
- Can be transparent and not alter the messages
- Non-transparent and alter the messages
- Can perform several functions including
- Caching (storing a copy of a given resource) can be private or public
- Filtering (like an antivirus scan/parental controls)
- Load balancing (spreading the requests across multiple servers)
- authentication (access control)
- Logging (storing historical info)

HTTP aspects

- **Simple:** simple and readable by a human, making it easier to test and reducing complexities for those new to HTTP
- **Extensible:** HTTP is easy to extend and experiment with, new functionality can be introduced by an agreement between a client and server
- **Stateless**: There is no link between two requests beings successively carried out on the same connection. At any time the client can send a valid command, server will not relate this command to any previous or future commands
- **Session-tracking**: Through the use of HTTP cookies, allowing stateful sessions. HTTP requests can share the same state/context with this
- **Connection:** Currently , HTTP relies on TCP

What can HTTP control

- **Caching:** how docs are cached is controlled by HTTP, the server can instruct proxies and clients what to cache and for how long. The client can instruct cache proxies to ignore the stored doc
- **Relax the original constraint**: Can relax the feature of only allowing scripts on different web pages from access other pages if they have the same origin
- **Authentication:** can protect web pages by only allowing specific user to access them
- **Sessions:** HTTP cookies allow linking requests with the state of the server, creating sessions allowing for user configuration of the output (also useful for shopping baskets)

HTTP Conclusion

- HTTP is extensible and easy to use.
- Uses Client/Server Architecture with headers in order to extend the capability of the web