Day1 - HTML

Client vs Server-Side

- Client can run only HTML, CSS, Javascript
- Typescript is complied to Javascript
- Server-side for security
- Storage in browser (local storage, cache)

IP address: v4 size 32 bit, v6 128

TCP = Transmission Control Protocol

Multiplexing

Multiple programs using the same IP address, by using a port, a number given to each program or service

- port 80: default
- port 443: for secure connection
- port 21: ftp
- port 22: ssh

DNS = Domain Name system

DNS servers map written names to IP addresses

Many systems maintain a local DNS cache called a host file:

- Windows: C:\Windows\system32\drivers\etc\hosts
- Mac: /etc/hosts
- Linus: /etc/hosts

HTTP = Hyper Text Transport Protocol

- HTTP **how messages are formatted**, and what actions Web servers and clients should take in **response to various commands**.
- Consists of header and body
- This is converted to object

HTML = Hyper Text Mark Language

- Content of HTTP repose body

<body> is converted to tree shaped object and it is also called DOM tree

- open/close markups (<p><\p>)
- self-closed markups(<img src="" \>)

HTTP Request

1. The client sends an HTTP request to the server
2. The server sends a response back

HTTP Verbs

- `GET` Retrieves data from the server
- `HEAD` Same as GET, but response comes without the body
- `POST` Submits data to the server
- `PUT` Replace data on the server
- `PATCH` Partially update a certain data on the server
- `DELETE` Delete data from the server

- `OPTIONS` Handshaking and retrieves the capabilities of the server (Preflight request)

URL = Uniform Resource Locator

- Anchor
  - http://www.miu.edu/download/index.html#downloads
- Port
  - http://www.cs.miu.edu:8080/mscs/wap.txt
- Query string
  - http://www.google.com/search?q=miu&start=10

Block vs Inline Elements

A block element always starts on a new line and takes up the full available width (stretches out to the left and right as far as it can).

Block elements = div

Inline elements = span

Demo: div, span, p, h1/6, hr, br, img, a, section, header, footer, nav, aside, article, ol/li, ul/li, table/tr/th/td, strong, i, <!-- comment -->

These are semantic. Each elements have their own uses and they should be used by their uses and other elements may not fit it.

Block Elements

<address><article><aside><blockquote><canvas><dd> <dl><dt>
<fieldset><figcaption><figure><footer><form><h1>-<h6><header><hr>
<main><nav><noscript>
<p><pre><section><table><tfoot>
<video>

Inline Elements

<a><abbr><acronym><b><bdo><big>
<button><cite><code><dfn>
<i><img><input><kbd><label><map><object><output><q><samp><script>
<select><small><sub><sup><textarea><time><tt><var>

Absolute vs Relative Path

Absolute URL:

A complete address containing all the parts needed to find a specific file or web page on the internet.

Relative URL:

It provides directions to find a resource relative to the current webpage. A relative URL skips some parts of the absolute URL, assuming the browser can fill in the gaps based on the current webpage location.

Notes:

Browser still have to convert relative path to absolute path and <base> helps to do it.

Browser still tries to use base by inferring the webpage URL.

helps to <base> ⇒ tag specifies the base URL and/or target for all relative URLs in a document.

Form Elements

Forms start with <form> and end with </form>, attributes include: action, method, enctype, novalidate, autocomplete

Form Elements include:

input, textarea, button, select/option, label

An input element must have a name, value, and type. Types include: text, checkbox, radio, file, password, number, date, time, email, time, color.. etc

A button element must have a type. Types include: button, submit, reset

Names, values and type