CS Study 7주차

google.com을 검색하면

김신아

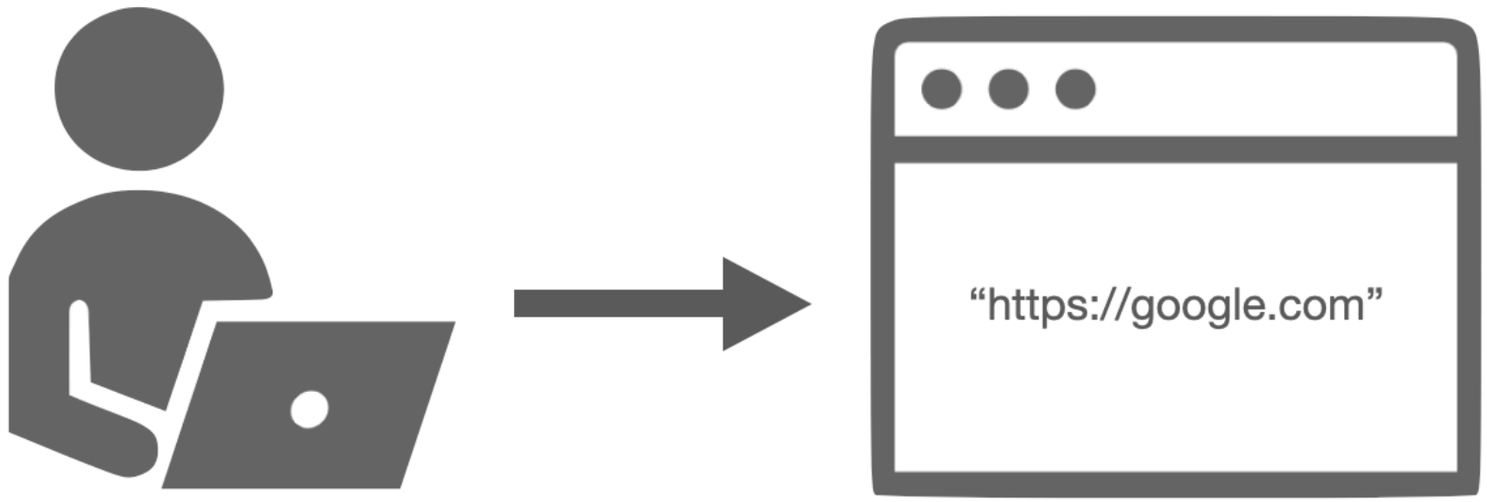
오늘 알아볼 내용은?

브라우저에서 “google.com"을 검색하면 일어나는 일

오늘 알아볼 내용은?

1. “google.com" 입력
2. 브라우저는 “google.com"에 상응하는 IP 주소를 찾기 위해 DNS 기록 캐시 확인
 2 - 1. 만약 요청한 URL이 캐시에 없다면 ISP의 DNS 서버는 “google.com"을 갖는 서버의 IP 주소를 찾기 위해 DNS query를 시작
3.URL 주소 중 도메인 네임 부분을 DNS 서버에서 검색
4. 브라우저는 서버와 TCP 통신 시작
5. 브라우저는 웹 서버에 HTTP 요청을 보냄
6. 서버는 요청을 처리하고 응답을 다시 보냄
7. 서버는 HTTP 응답을 내보냄
8. 브라우저는 HTML 컨텐츠 표시

1. google.com 검색



2. DNS (Domain Name System) 기록 캐시 확인

• **DNS (Domain Name System)**

: 웹사이트의 이름(URL)과 그것이 연결된 IP 주소를 갖고 있는 데이터베이스이다. 모든 URL은 자신만의 IP 주소를 가지고 있다. 이때 IP 주소는 우리가 접속하려하는 웹사이트의 서버를 가지고 있는 컴퓨터에 있다.

DNS 기록을 찾기 위해서 브라우저는 **browser -> OS -> router -> ISP**순으로 확인한다.

캐시는 네트워크 트래픽을 통제하고 데이터 전송 시간을 높이는데 필수적으로 필요하기 때문이다.

3. DNS (Domain Name System)

- **만약 요청한 URL이 캐시에 없다면 ISP의 DNS 서버는 “google.com"을 갖는 서버의 IP 주소를 찾기 위해 DNS query를 시작**
- **DNS (Domain Name System)**

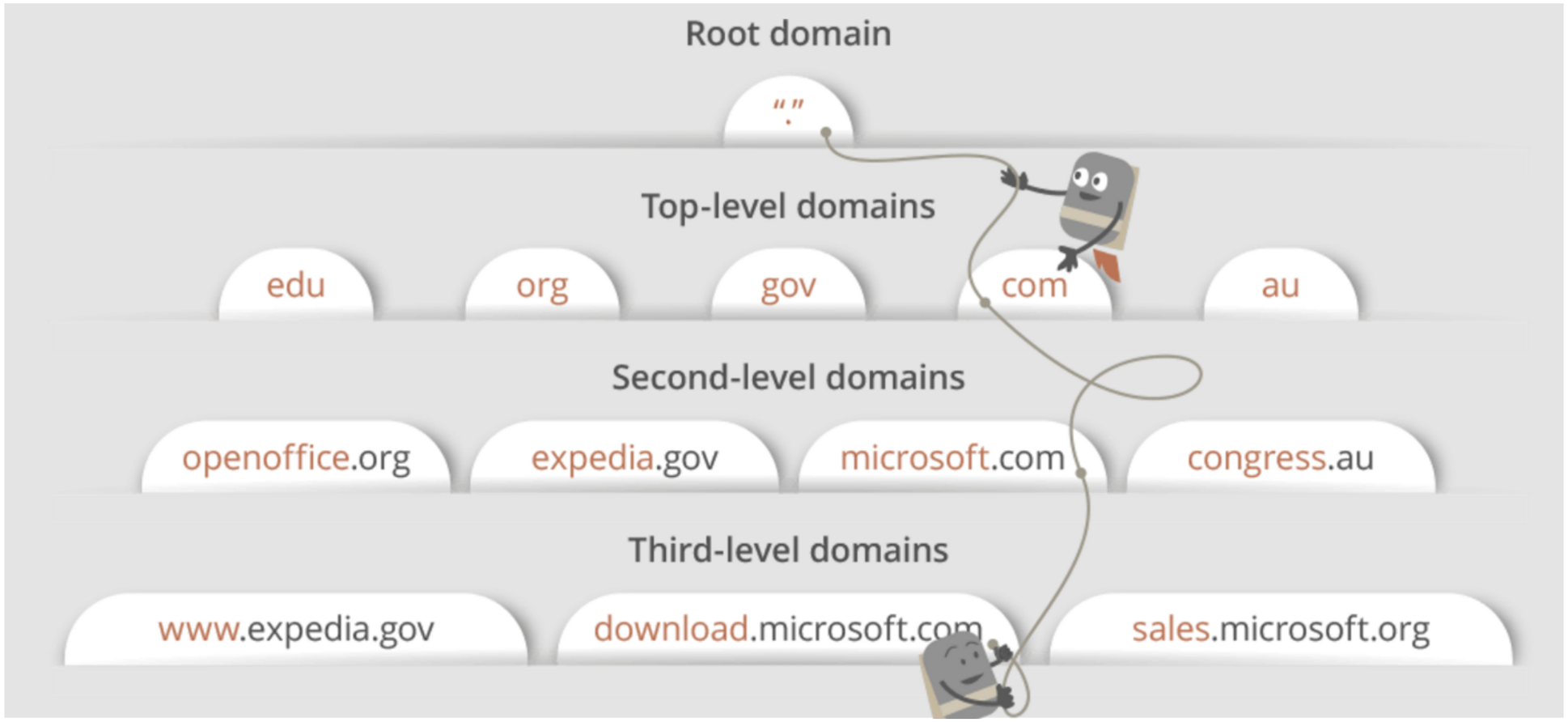
: 우리가 알고 있는 주소(google.com)를 IP 주소로 변환하는 과정에 개입한다. 점(.)으로 구분되며 계층적으로 구성되어 있다. 계층 끝은 root로 root dns 서버가 나올 때까지 거꾸로 타고 올라간다.

DNS 프로토콜은 내 서버와 가장 가까운 DNS 서버는 한번 조회된 주소에 대해서는 어느 정도의 기간동안 보관할 것인지 DNS 프로토콜에 정의해야한다. 일정 이하의 TTL(Time-to-Live)은 무시함 실무적으로는 보통 3000초정도이다.

3. DNS (Domain Name System)

- **DNS (Domain Name System) query**

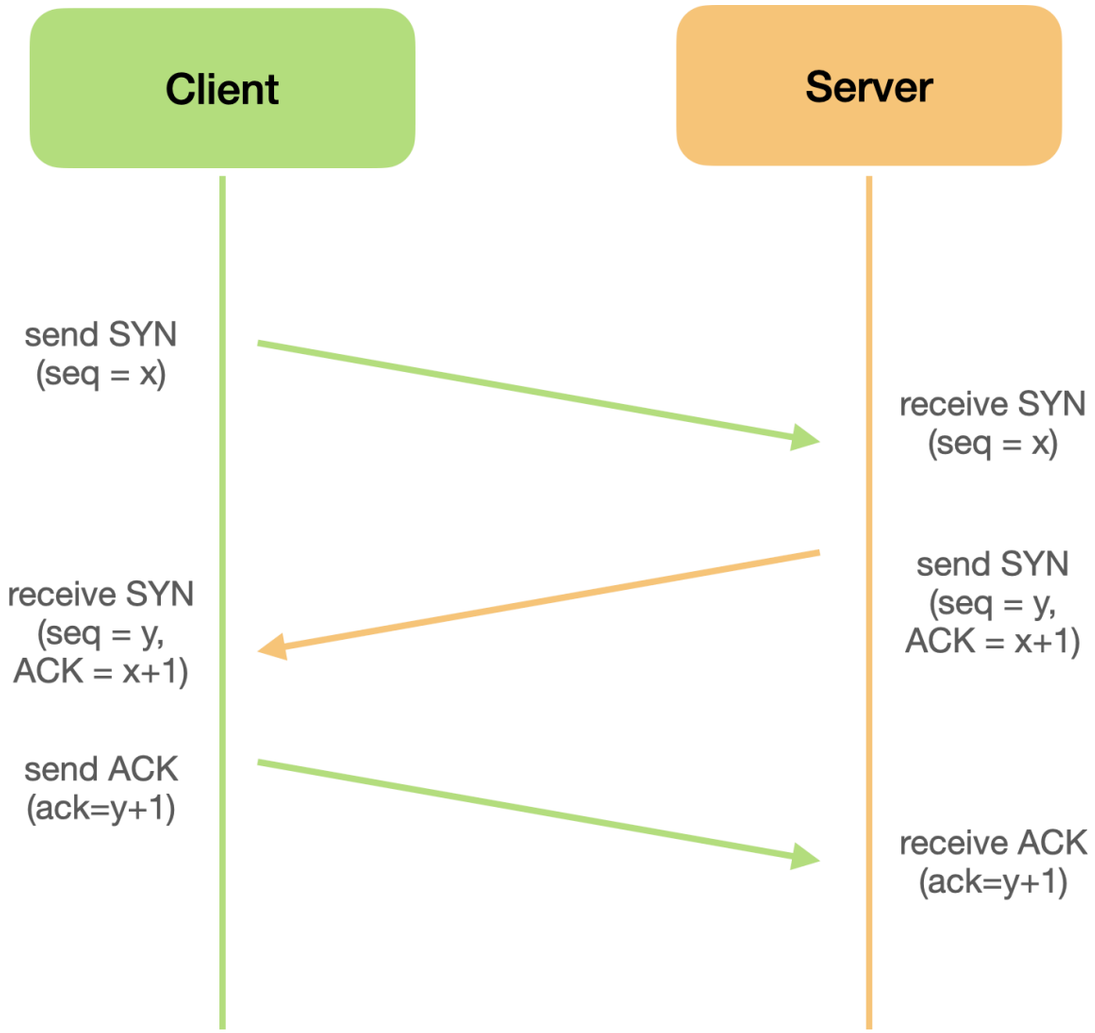
DNS query의 목적은 올바른 IP 주소를 찾을 때까지 인터넷에 많은 DNS server를 검색하는 것이다. 이런식의 검색을 recursive search(반복되는 검색)라고 한다.
이러한 상황에서 ISP의 DNS server를 DNS recursor라고 부른다. 다른 DNS server는 name server라고 부르는데 웹사이트의 도메인 이름의 구조를 기반으로해서 DNS 검색을 수행하기 때문이다. 우리가 접하는 URL은 대부분 third-level, second-level, top-level 도메인을 가지고 있고, 각 도메인은 DNS 검색 과정 동안 쿼리하는 자신만의 name server를 가진다.



4. 브라우저는 서버와 TCP 통신 시작

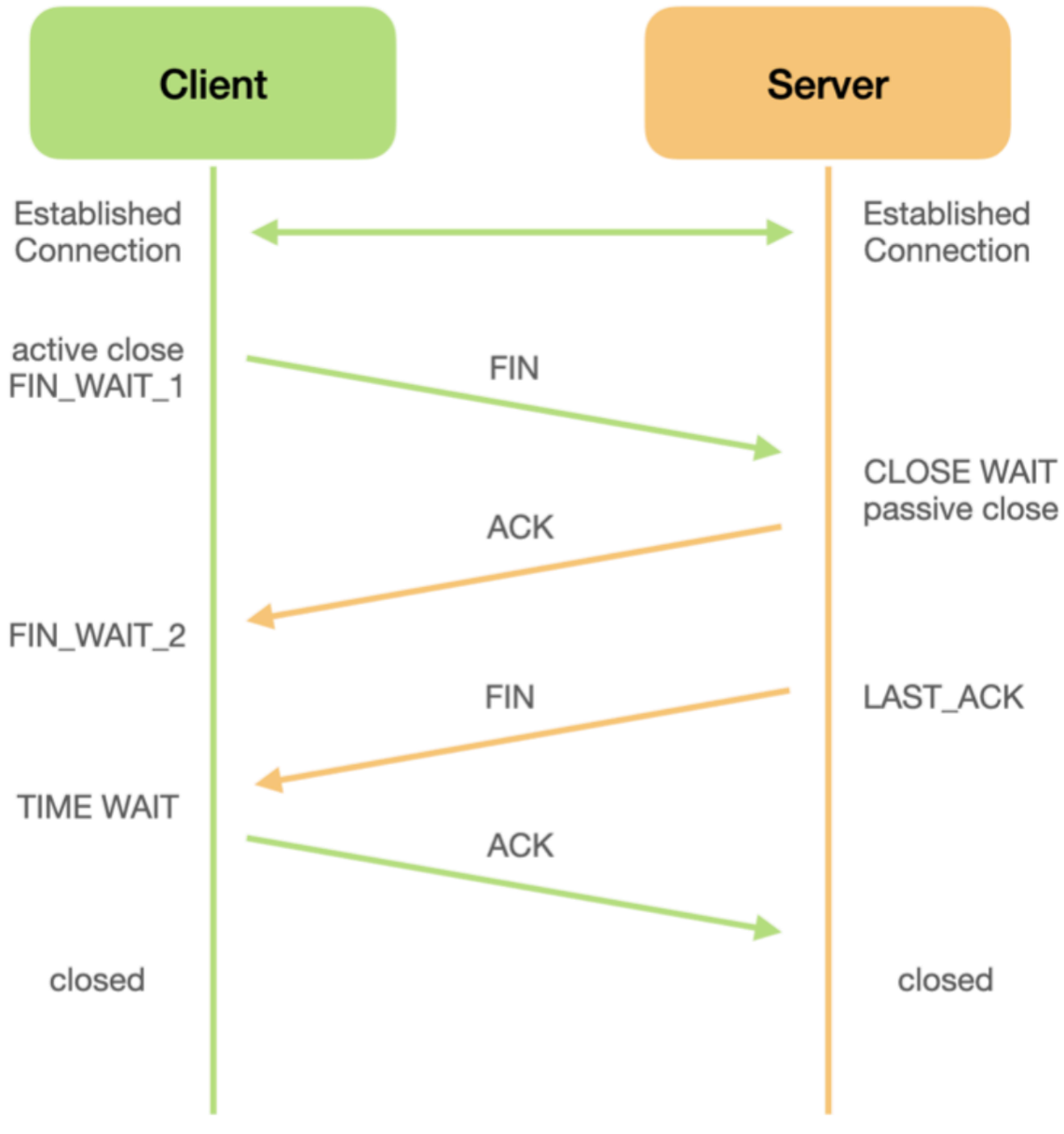
브라우저가 올바른 IP 주소를 받으면, IP 주소가 일치하는 서버와 정보 전달을 위한 연결 시도
브라우저는 연결을 위해 인터넷 프로토콜 사용, 여기에 사용되는 인터넷 프로토콜 중 TCP가 HTTP 요청을 위해 사용되는 가장 흔한 방법

- **TCP - 3 way handshake (연결 성립)**



질문 타임마~

- TCP - 4 way handshake (연결 해제)



5. 브라우저는 웹 서버에 HTTP 요청을 보냄

• **HTTP(Hyper Text Transfer Protocol)**

: TCP 기반의 클라이언트와 서버 사이에 이루어지는 요청/응답 프로토콜

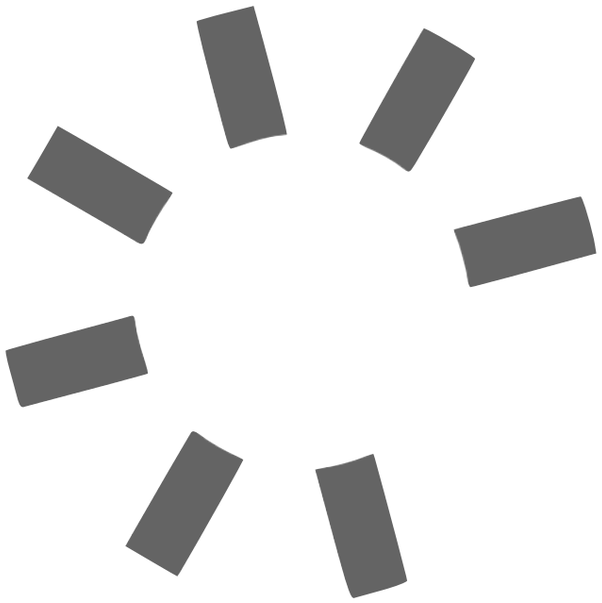
HTTP는 Text Protocol로 사람이 쉽게 읽고 쓸수 있다. 프로토콜 설계상 클라이언트가 요청을 보내면 반드시 응답을 받아야 한다. 응답을 받아야 다음 request를 보낼 수 있다.

브라우저 식별정보(User-Agent), 수락에 대한 요청(Accept header), TCP 연결을 유지하도록 요청하는 Connection header 등에 대한 추가적인 정보를 가질 수 있다. 브라우저가 도메인 저장소에 갖는 쿠키로부터 가져온 정보를 넘길수도 있다.

```
GET http:// google.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: google.com
Cookie: datr=1265876274-[...]; locale=en_US; lsd=WW[...]; c_user=2101[...]
```

- GET Request -

6. 서버는 요청을 처리하고 응답을 다시 보냄



7. 서버는 HTTP 응답을 내보냄

- **HTTP server 응답 예시**

서버 응답은 요청한 페이지와 함께 status code, compression type(Content-Encoding), 페이지를 캐시하는 법(Cache-Control), 프라이빗 정보 등을 담고 있다.

```
[➜  ~ telnet www.google.com 80                                                  ]
Trying 172.217.26.36...
Connected to www.google.com.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.google.com

HTTP/1.1 200 OK
Date: Sun, 13 Sep 2020 19:12:37 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Server: gws
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
Set-Cookie: 1P_JAR=2020-09-13-19; expires=Tue, 13-Oct-2020 19:12:37 GMT; path=/;
 domain=.google.com; Secure
Set-Cookie: NID=204=Z7wVe8VYXTPI5-WAKqdObH1dbMS0ENPOpmxH0v5P73i4xkRFRf1aq_zdCQSO
YxnYBZhj5cXex8T3Sw3NOzutX8SsFkqLfvnDm7Gems1o6m6NgSC5oReId9amyfSSicyQ0XiuphlJWT9i
WQ9UdZizVgdZ6bktNbztu5nSjV5tmg0; expires=Mon, 15-Mar-2021 19:12:37 GMT; path=/;
domain=.google.com; HttpOnly
Accept-Ranges: none
Vary: Accept-Encoding
Transfer-Encoding: chunked

515f
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="ko"
><head><meta content="text/html; charset=UTF-8" http-equiv="Content-Type"><meta
content="/images/branding/googleg/1x/googleg_standard_color_128dp.png" itemprop=
"image"><title>Google</title><script nonce="wbLdLOxKAzHdBosHnHvosg==">(function(
```

- GET Response -

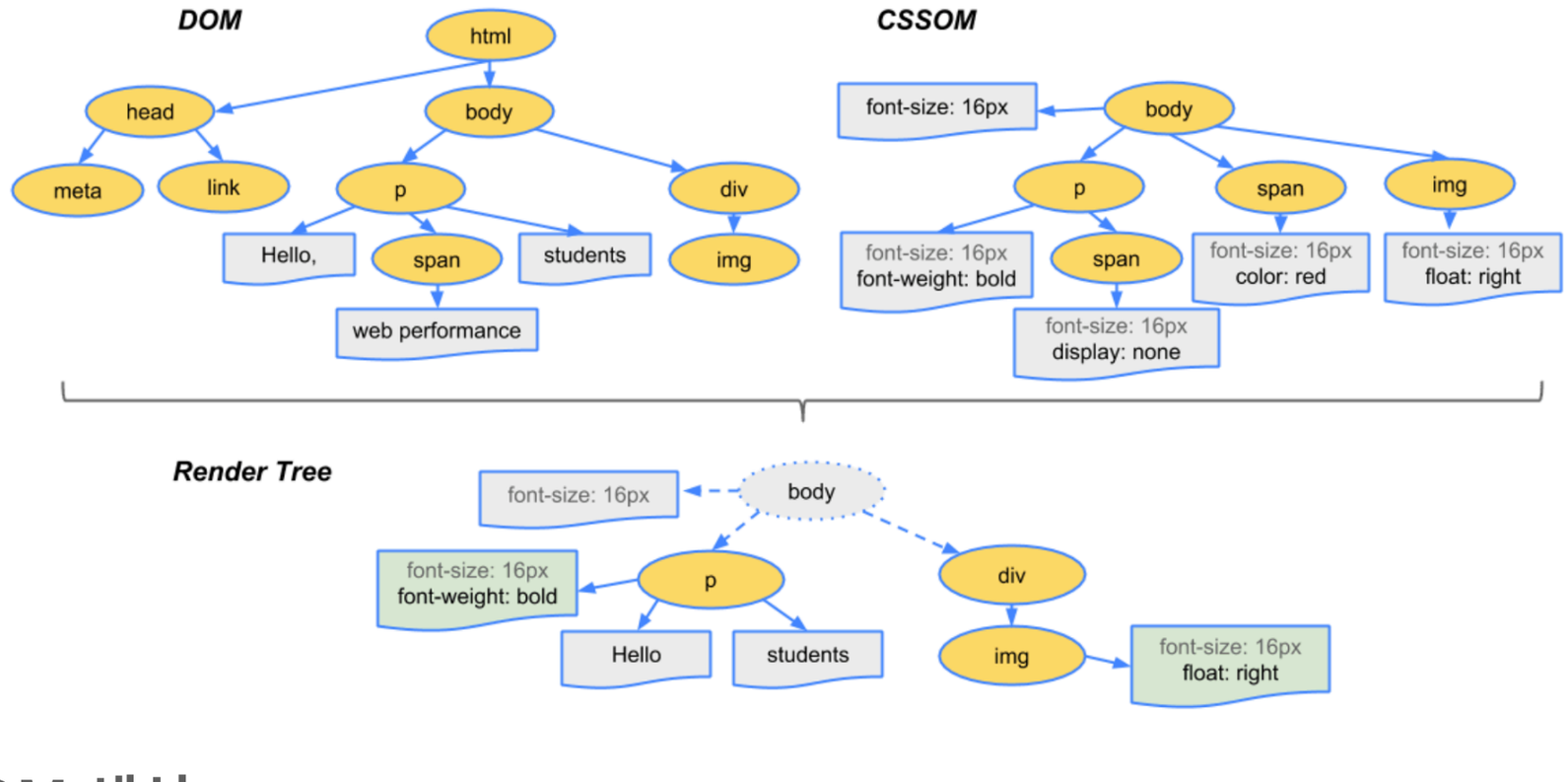
8. 브라우저는 HTML 컨텐츠 표시

• **HTML 컨텐츠 표시**

브라우저에서 Content-type이 text/html 형식의 문서를 받았다고 한다면, 이를 파싱하여 **렌더링**하는 과정을 거친다.

• **렌더링 ->>>>>**

8. 브라우저는 HTML 컨텐츠 표시



• DOM과 CSSOM 생성

브라우저가 페이지를 렌더링하려면 먼저 DOM 및 CSSOM 트리를 생성해야 한다.

• DOM(Document Object Model)

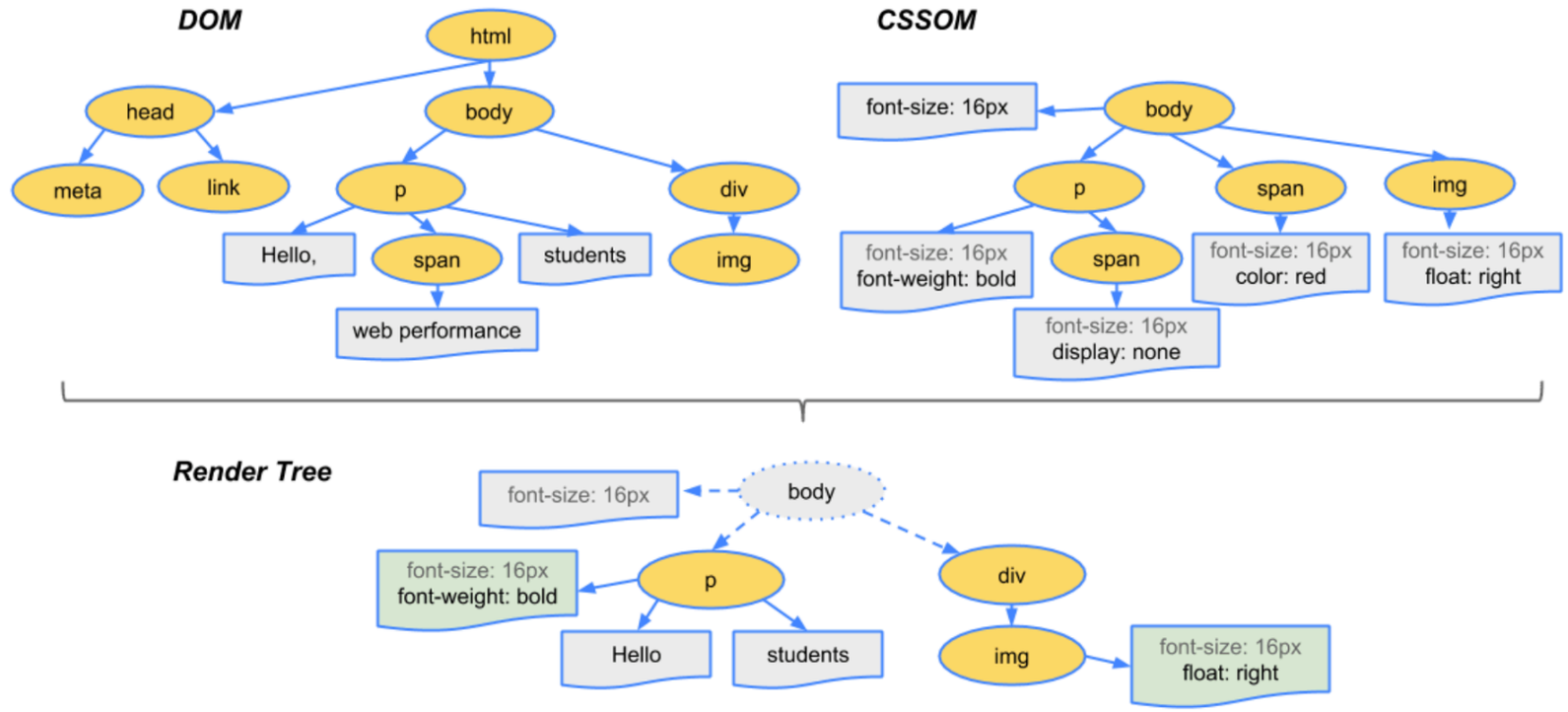
브라우저가 HTML 파일을 읽을 때 브라우저가 이해할 수 있는 메모리에 보관할 수 있는 Object로 변환시킨다.

• CSSOM(CSS Object Model)

브라우저는 HTML 파일을 DOM으로 만들면서 우리가 정의한 CSS뿐만 아니라 브라우저에서 기본적으로 설정되어 있는 CSS를 cascading룰에 따라 병합하여 CSSOM을 만든다.

DOM + CSS -> CSSOM

8. 브라우저는 HTML 컨텐츠 표시



• **Render Tree 생성**

DOM Tree와 CSSOM Tree를 결합하여 Render Tree를 형성한다. 요소들의 구조와 텍스트만 있는 DOM과 달리, 스타일 정보가 설정되어 있으며, Render Tree에는 페이지를 렌더링하는데 필요한 노드만 포함된다.

DOM + CSSOM -> Render Tree

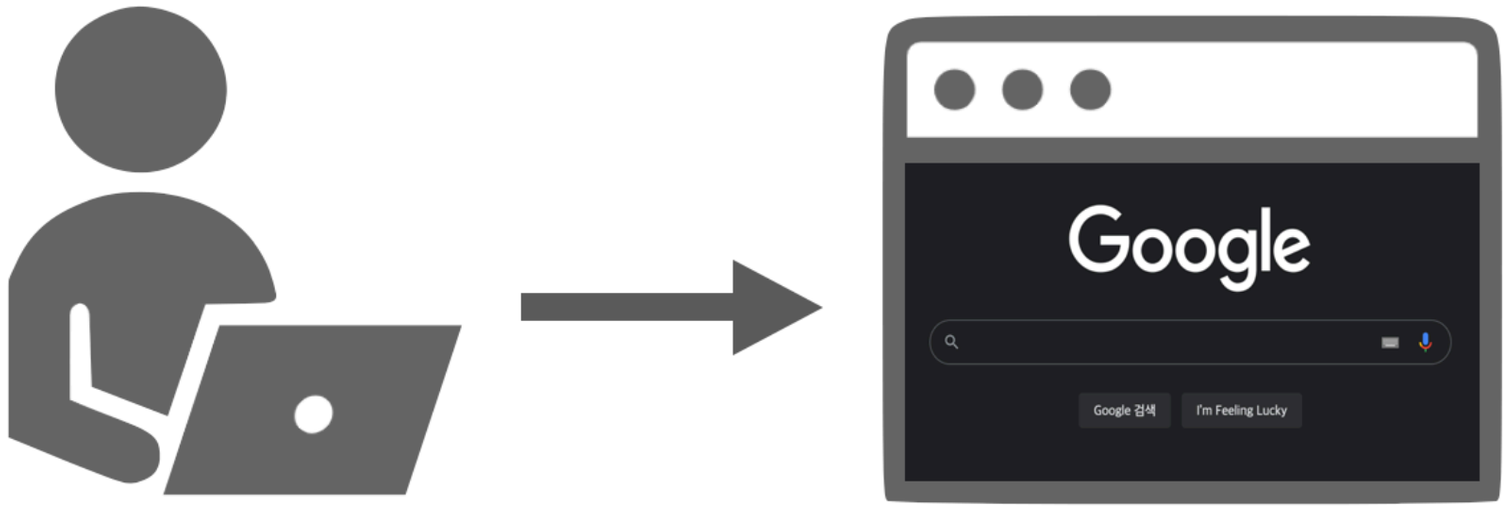
• **Layout**

각 객체의 정확한 위치와 크기를 계산한다.

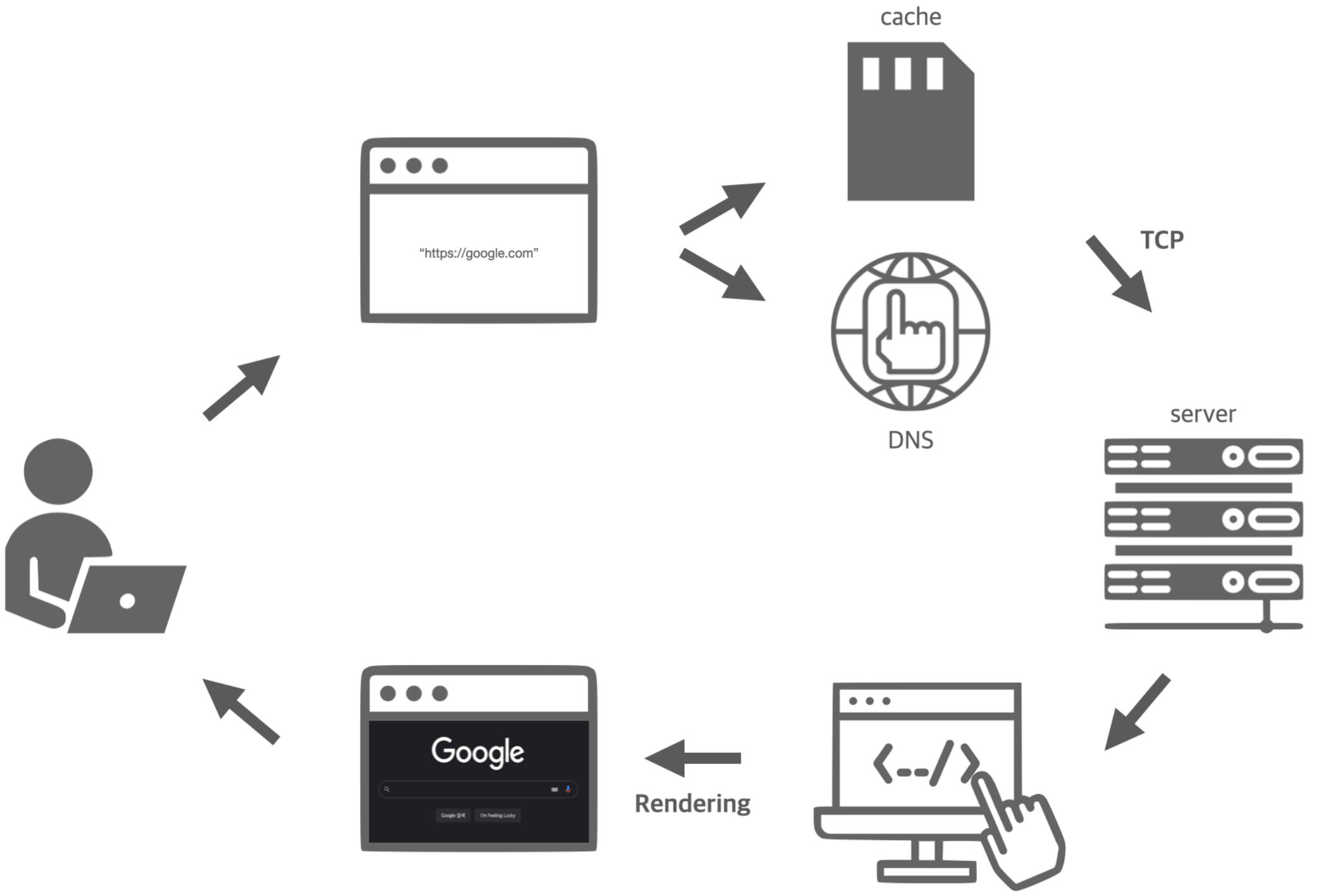
• **Paint**

최종 렌더링 트리에서 수행되는 페인트이며, 픽셀을 화면에 렌더링한다.

끝



최종 정리


cache
"https://google.com"
TCP
server
DNS
Google
Google 검색
I'm Feeling Lucky
Rendering

Thank You

SSL / TLS

• **SSL (Secure Sockets Layer) TLS (Transport Layer Security)**

: 컴퓨터 네트워크에 통신 보안을 제공하기 위해 설계된 암호 프로토콜

TCP 계층까지는 본질적으로 암호화등의 보안 기능을 제공하지 않는다. 보안 통신을 위해서는 암호화 등의 보안 기능을 제공하는 레이어가 필요한데, 이를 위한 기능을 제공하는 레이어이다.

SSL/TLS를 사용한 보안 프로토콜 중 가장 대표적인 것이 HTTPS이다.

인터넷 계층

• **인터넷 계층**

: 인터넷 작업에서 필요한 것이 어드레싱과 라우팅이다. 이 두가지를 수행해서 TCP/IP로 인터넷 작업을 수행하기 위한 프로토콜이 IP이다.

• **어드레싱과 라우팅**

: 어드레싱은 어드레스를 어떻게 써서 어떻게 배당하는 지에 대한 작업, IP 주소 지정 작업을 한다. 데이터를 목적지까지 전달하기 위해서는 라우터라고 하는 네트워크 장비가 필요하다.

• **IP**

: 비신뢰성, 비연결지향 데이터그램 프로토콜로 패킷으 받아서 주소를 해석하고 경로를 결정하여 다음 호스트로 전송하는 역할을 한다.

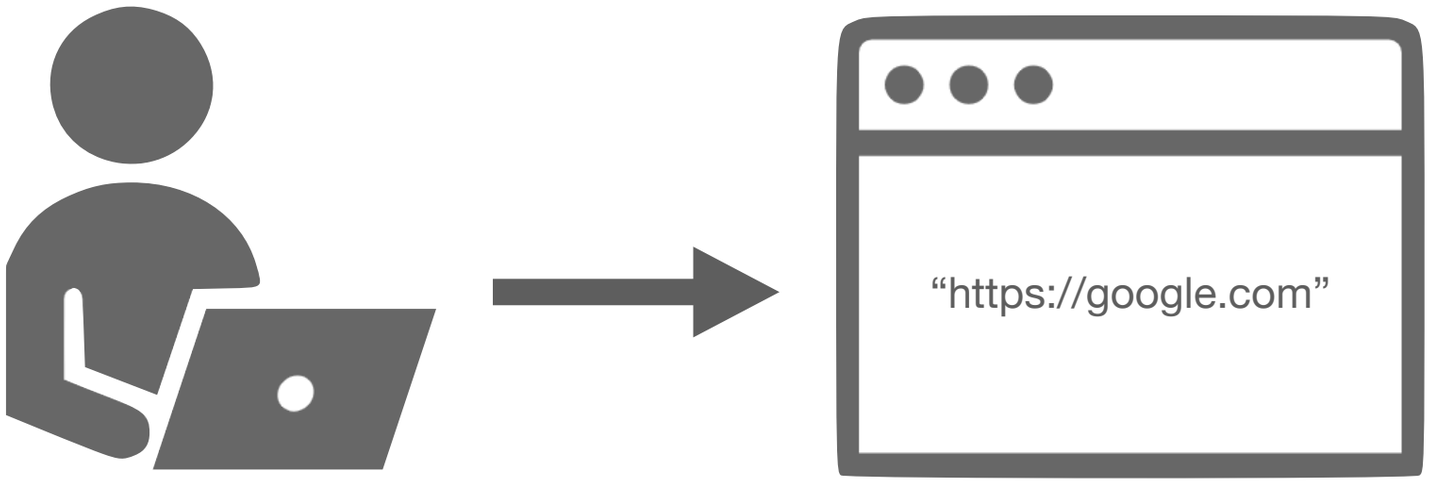
인터페이스 계층

• **인터페이스 계층**

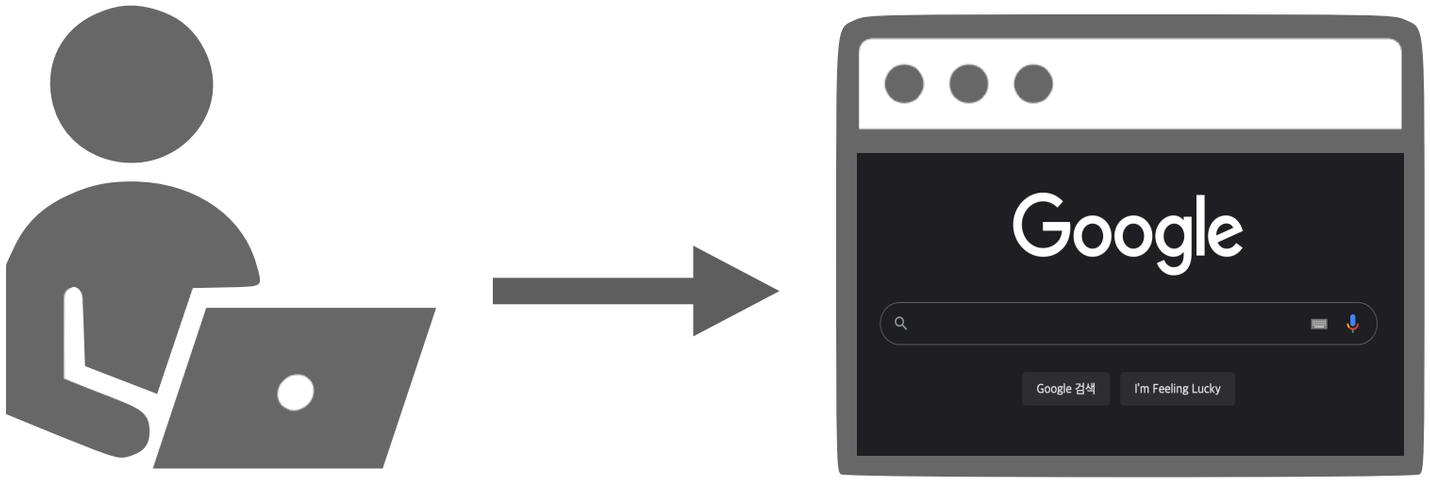
: 무선 LAN, 유선 LAN 등 하드웨어들을 제어하면서 인접한 다른 통신 기기까지 데이터를 전달하는 역할을 한다. 우리가 보내려고 하는 데이터들을 실제로 전송하는 계층이다.

위와 같은 과정을 거쳐 리소스를 요청하면 서버에서 Response를 생성하여 브라우저에게 응답한다. 브라우저는 이 문서를 파싱하여 처리를 한다.

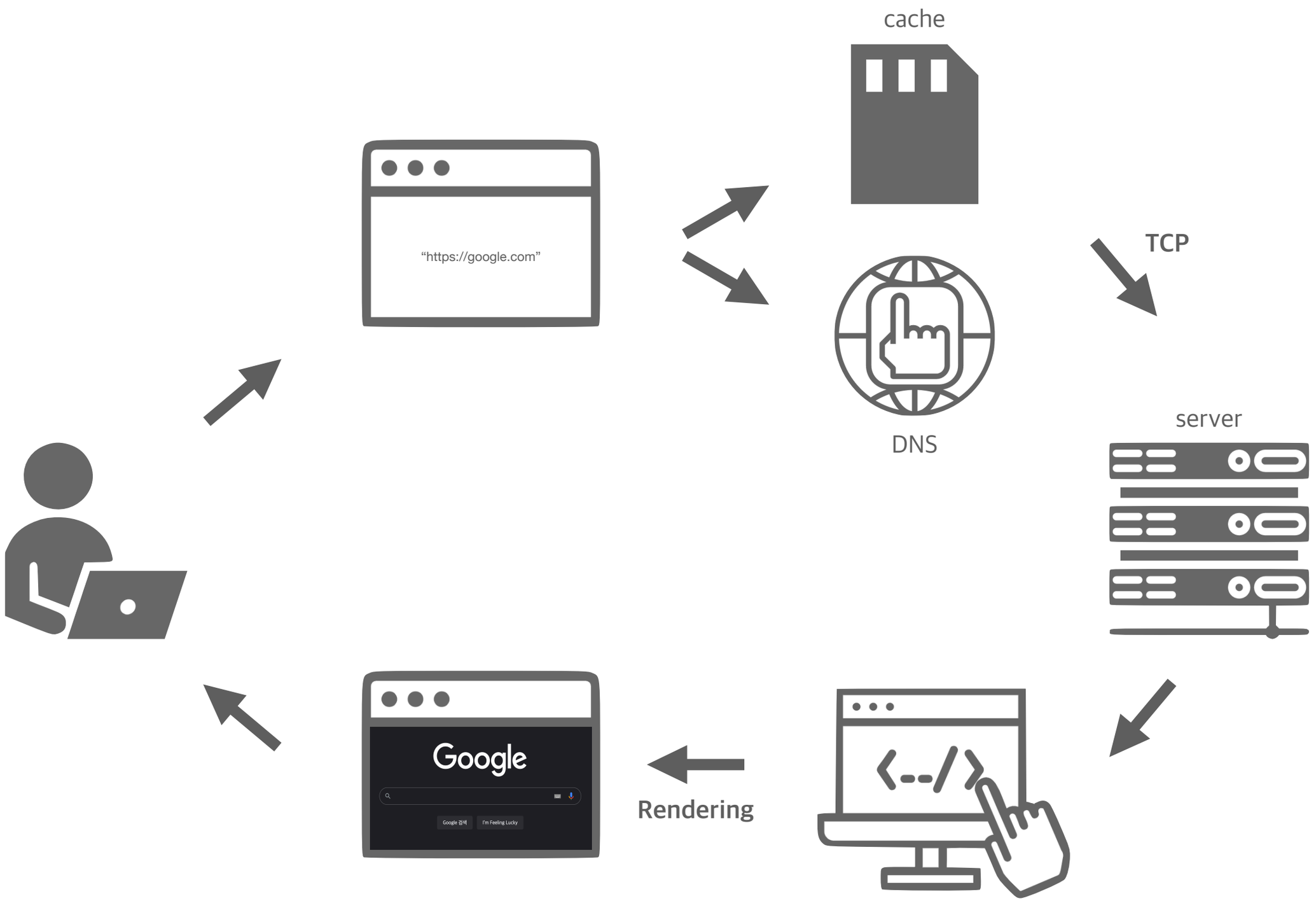
별첨



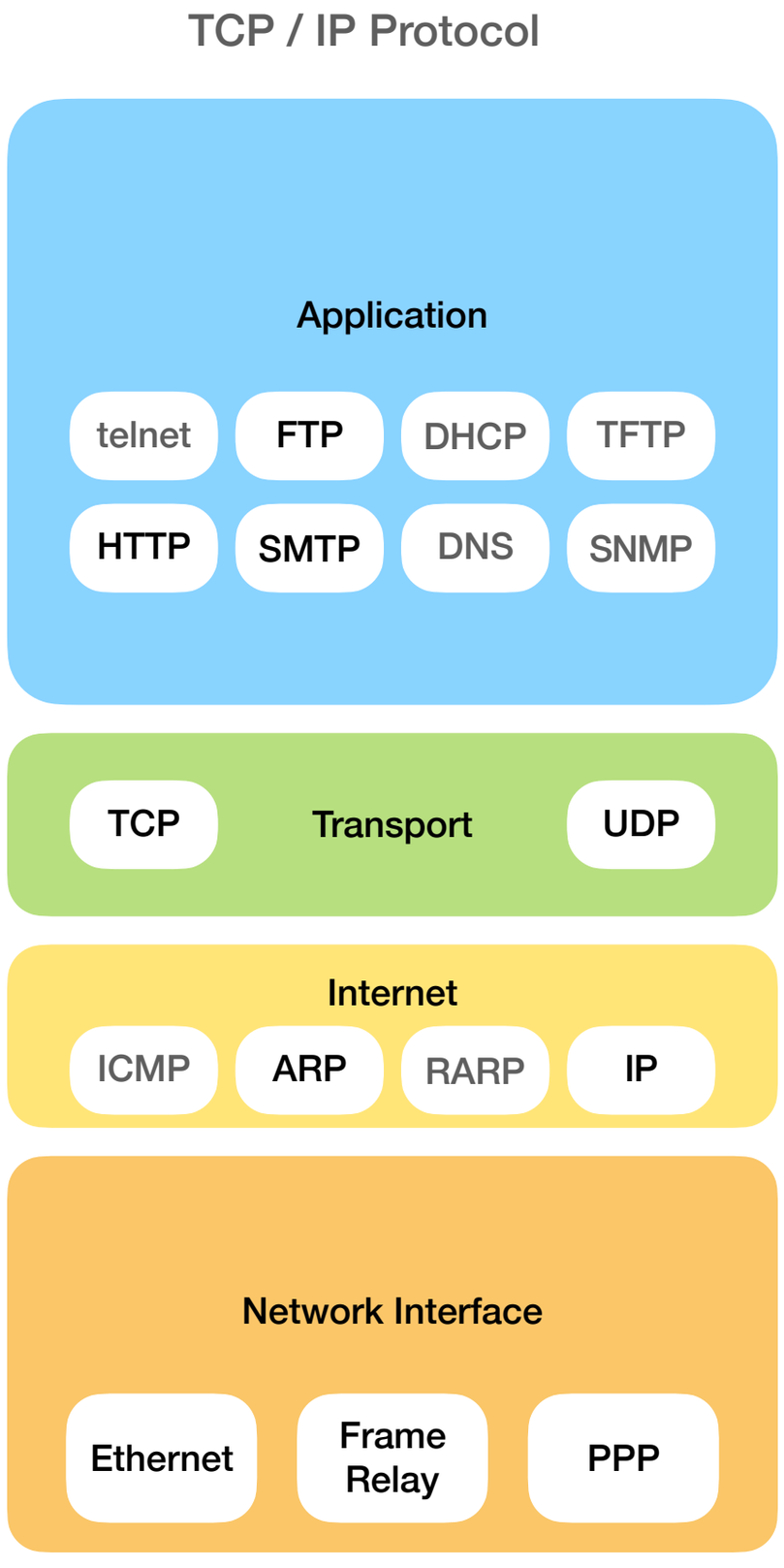
별첨



별첨


cache
"https://google.com"
TCP
server
DNS
Google
Google 검색
I'm Feeling Lucky
Rendering

별첨


TCP / IP Protocol
Application
telnet
FTP
DHCP
TFTP
HTTP
SMTP
DNS
SNMP
TCP
Transport
UDP
Internet
ICMP
ARP
RARP
IP
Network Interface
Ethernet
Frame Relay
PPP

별첨

https://velog.io/@jjo-niixx/%EB%B8%8C%EB%9D%BC%EC%9A%B0%EC%A0%80%EC%97%90-URL-%EC%9E%85%EB%A0%A
%ED%9B%84-%EC%9D%BC%EC%96%B4%EB%82%98%EB%8A%94-%EC%9D%BC

https://bohyeon-n.github.io/deploy/network/internet-2.html

https://designer-ej.tistory.com/entry/Web-Rendering-DOMCSSOMRender-Tree