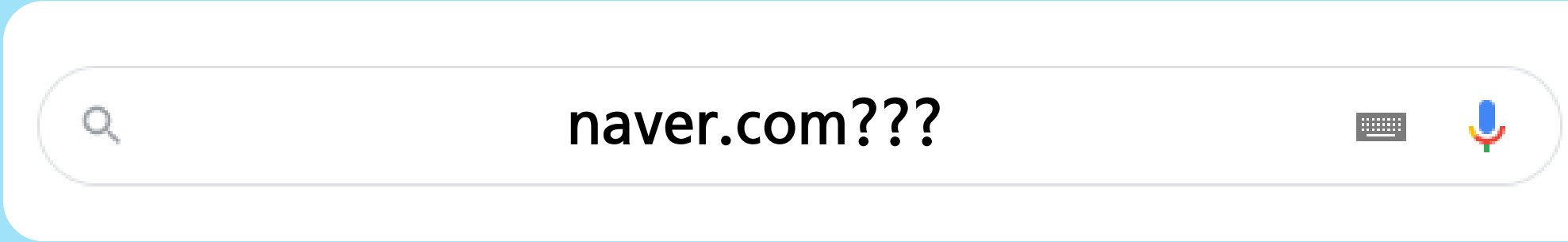


2021. 01.12 ImHoJeong

TL:DR;

브라우저는 렌더링 엔진이다!

웹 페이지를 다운로드 받아 사람이 이해할 수 있는 방식으로 렌더링을 해줍니다.

사용자가 주소 창에 주소를 입력합니다.

브라우저가 URL에 있는 문서 다운로드하고 렌더링을 합니다

브라우저가 하는 주요 기능

1. DNS resolution (DNS lookup)

2. HTTP exchange

3. Rendering

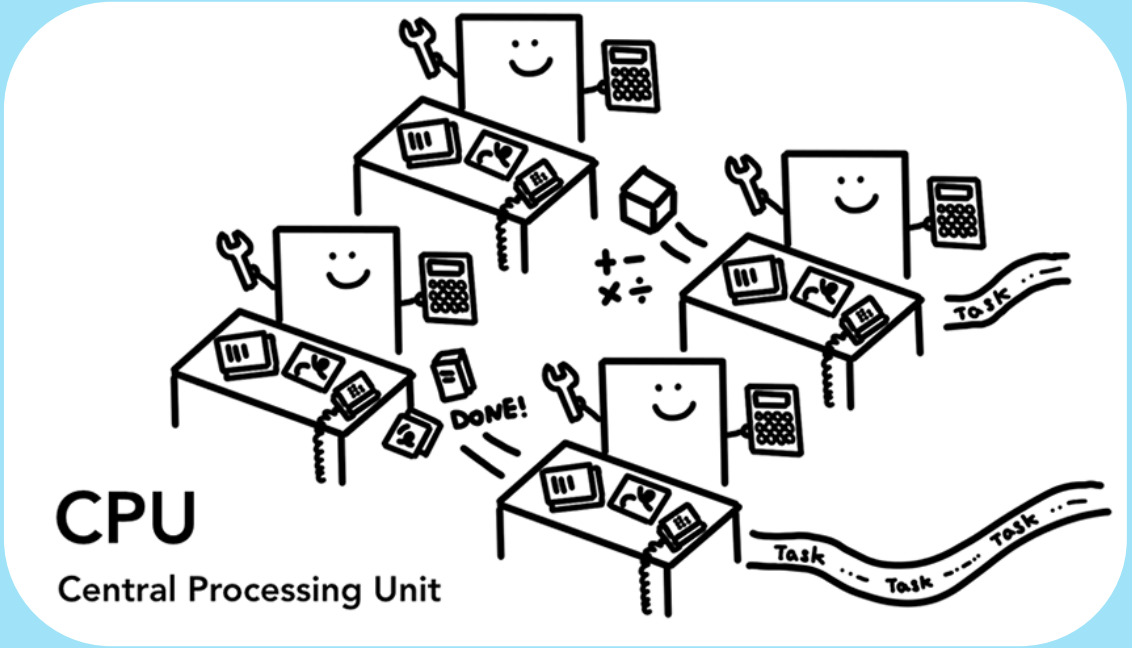
4. Rinse and repeat - 무한 반복

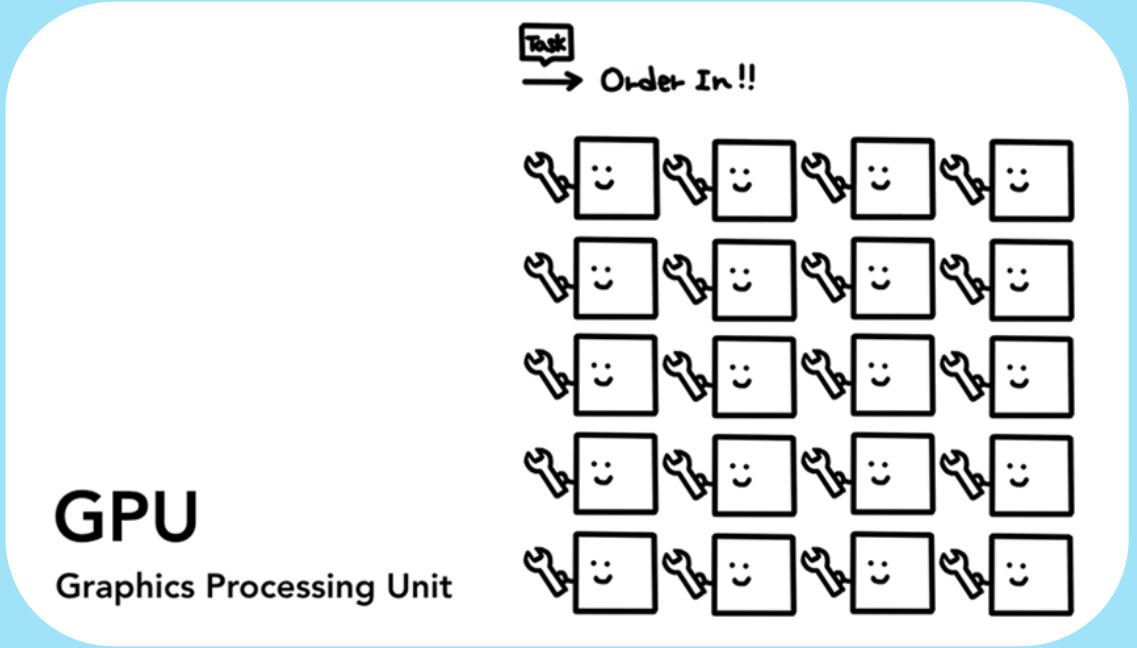
참고 : https://www.freecodecamp.org/news/web-application-security-understanding-the-browser-5305ed2f1dac/

modern Web Browser??

modern Web Browser - CPU & GPU

컴퓨터나 스마트폰에서 어플리케이션 시작 => CPU와 GPU가 앱을 실행





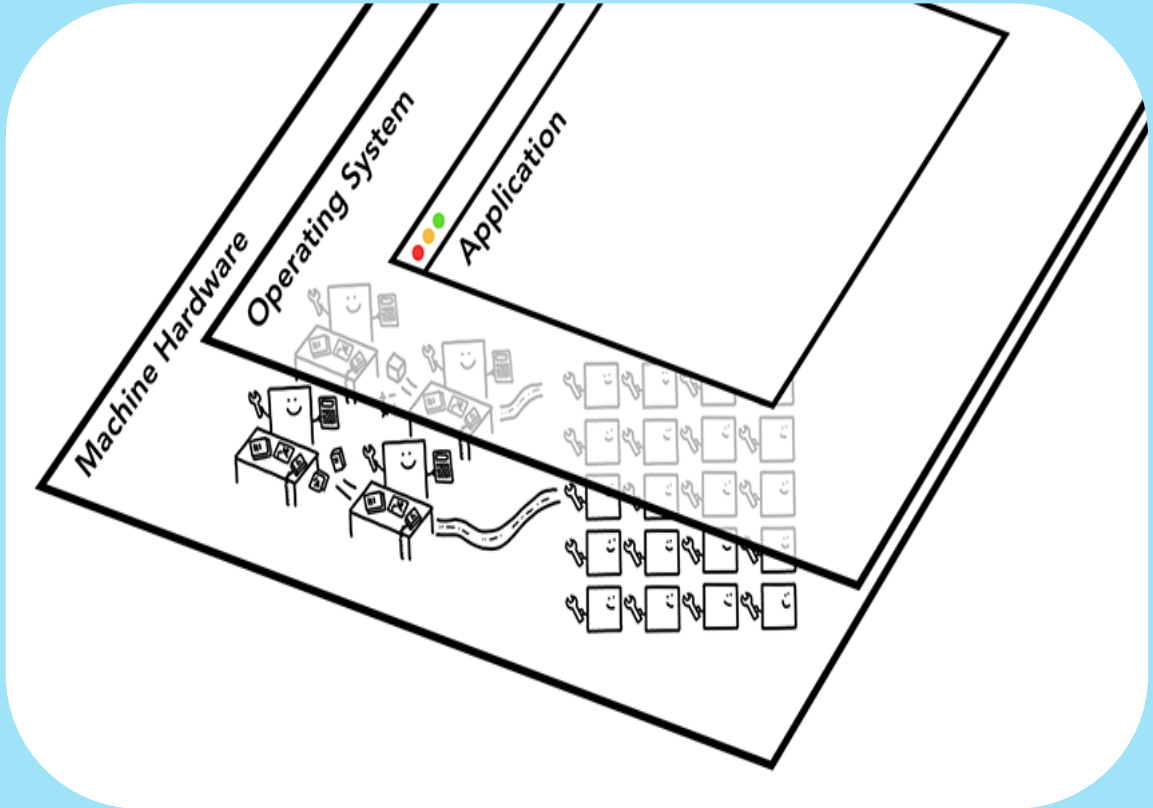
CPU

매우 다양한 작업들을 들어올 때마다
하나씩 처리

GPU

간단한 작업을 수 많은 코어에서
동시에 처리하는 데 특화

modern Web Browser - Process & Thread



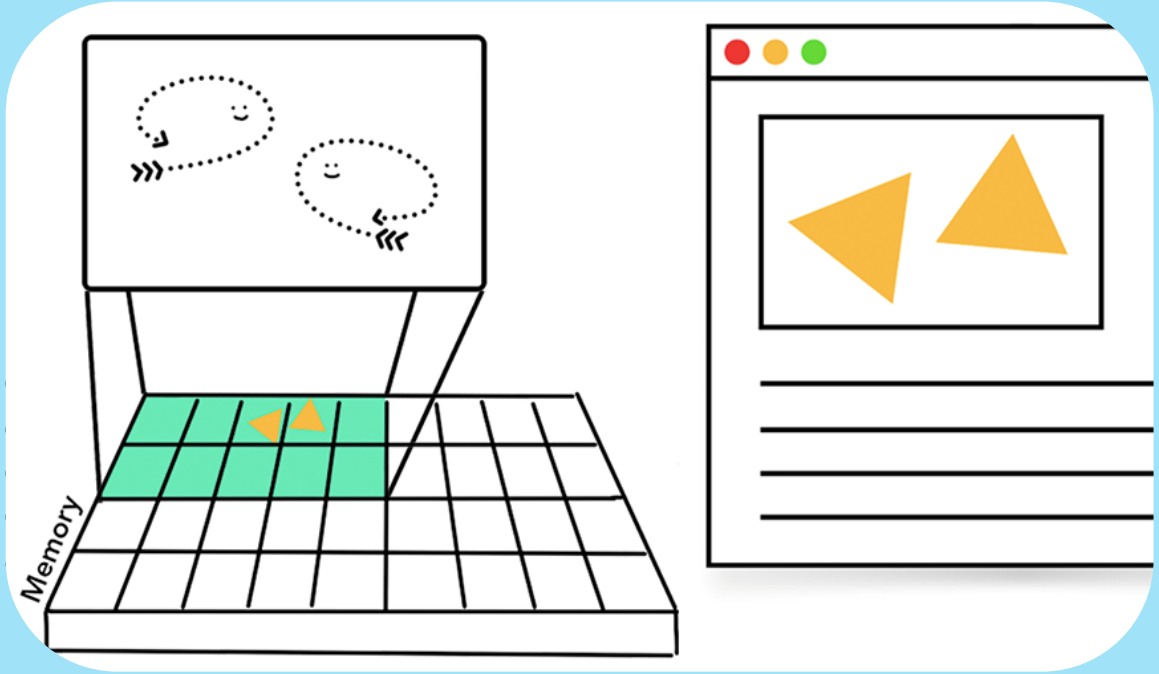
Process

어플리케이션의 실행 프로그램

Thread

프로세스 내부에 있으며,
프로세스의 프로그램을 실행하는 주체

1. 어플리케이션 시작 =〉 프로세스가 생성



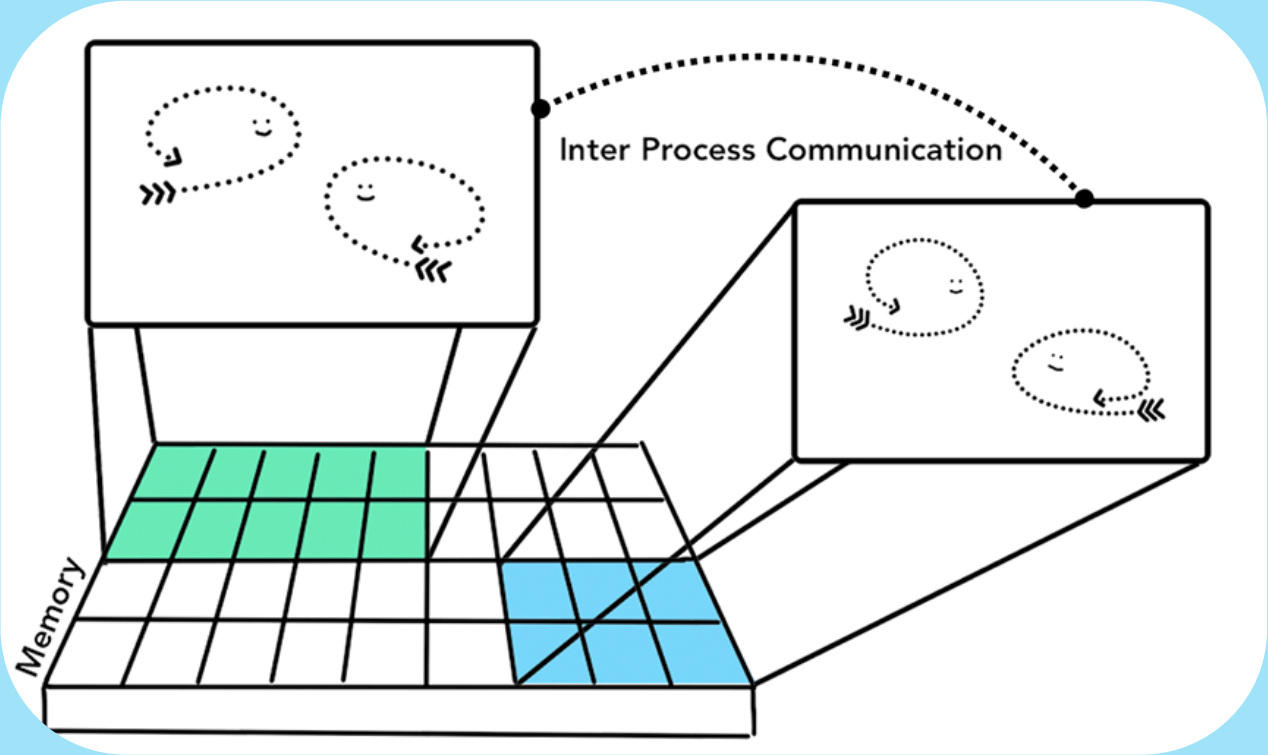
프로그램은 작업을 위해 스레드(들)을 생성할 수도 있음

OS는 프로세스에 메모리 한 조각을 주어서,
어플리케이션의 모든 상태정보를 고유 메모리
공간에 저장할 수 있게 함

어플리케이션을 종료하면 프로세스도 사라지고,
OS가 메모리를 해제

참고 : https://developers.google.com/web/updates/2018/09/inside-browser-part1?hl=ko

2. 프로세스는 다른 프로세스를 돌려서 별도의 작업을 수행하도록 OS에 요청할 수 있음



OS는 별도의 메모리 공간을 새 프로세스에 할당

두 프로세스 간 통신이 필요하다면 Inter Process Communication(IPC)를 이용

워커 프로세스가 무응답 상태에 빠지더라도 어플리케이션의 다른 부분을 수행하고 있는 프로세스들을 종료할 필요 없이 해당 프로세스만 재시작 가능

웹 브라우저도 한 프로세스가 스레드를 왕창 들고 있거나,
스레드 몇 개 가진 다수의 프로세스들이 IPC를 통해 통신

참고 : https://developers.google.com/web/updates/2018/09/inside-browser-part1?hl=ko

Review : 브라우저가 하는 주요 기능

서버로 요청한 리소스를 브라우저 화면에 보여주는 것

리소스 주소는 URI(Uniform Resource Identifier)로 정해짐

사용자가 URL(web address)를 입력

브라우저는 문서를 가져와 그려줌(fetch & Render)

브라우저의 사용자 인터페이스

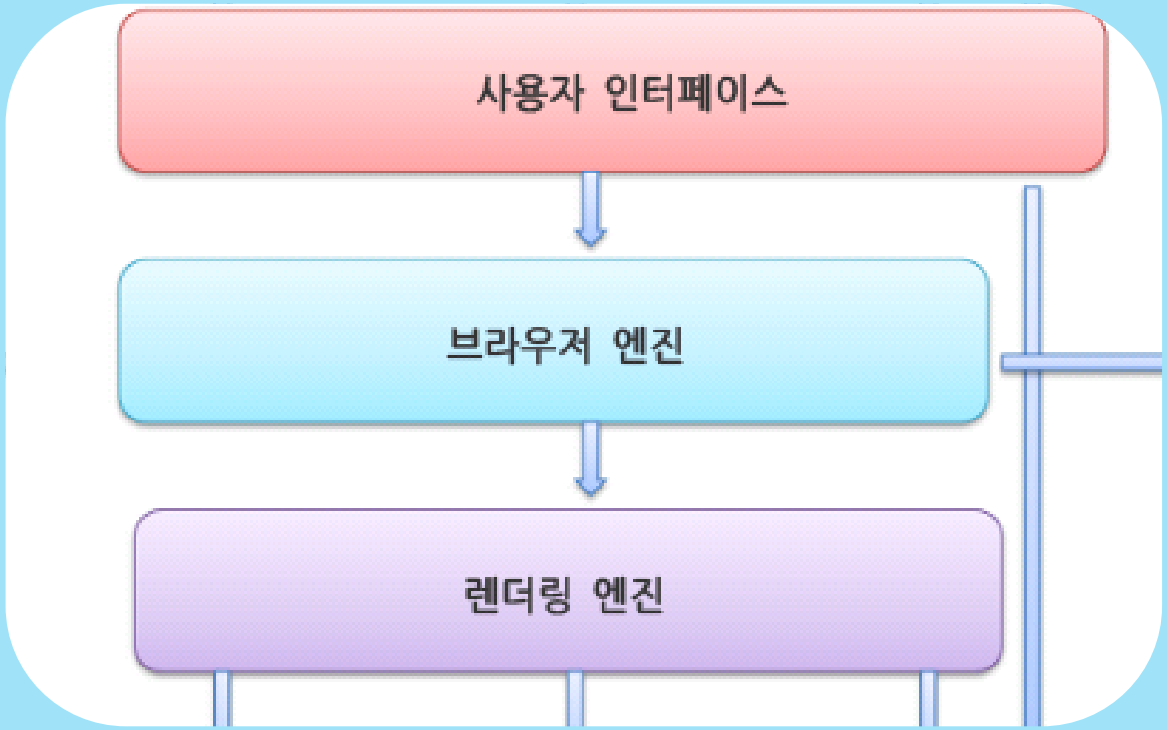
서로 닮아 있는데 일반적인 요소로는....

URL를 입력할 수 있는 주소 표시줄

이전 버튼 & 다음 버튼, 북마크, 홈 버튼

새로 고침 버튼 & 현재 문서의 로드를 중단할 수 있는 정지 버튼

브라우저의 기본 구조(1)



1) 사용자 인터페이스

주소 표시줄, 이전/다음 버튼, 북마크 메뉴 등
요청한 페이지를 보여주는 창을 제외한 나머지 모든 부분

2) 브라우저 엔진

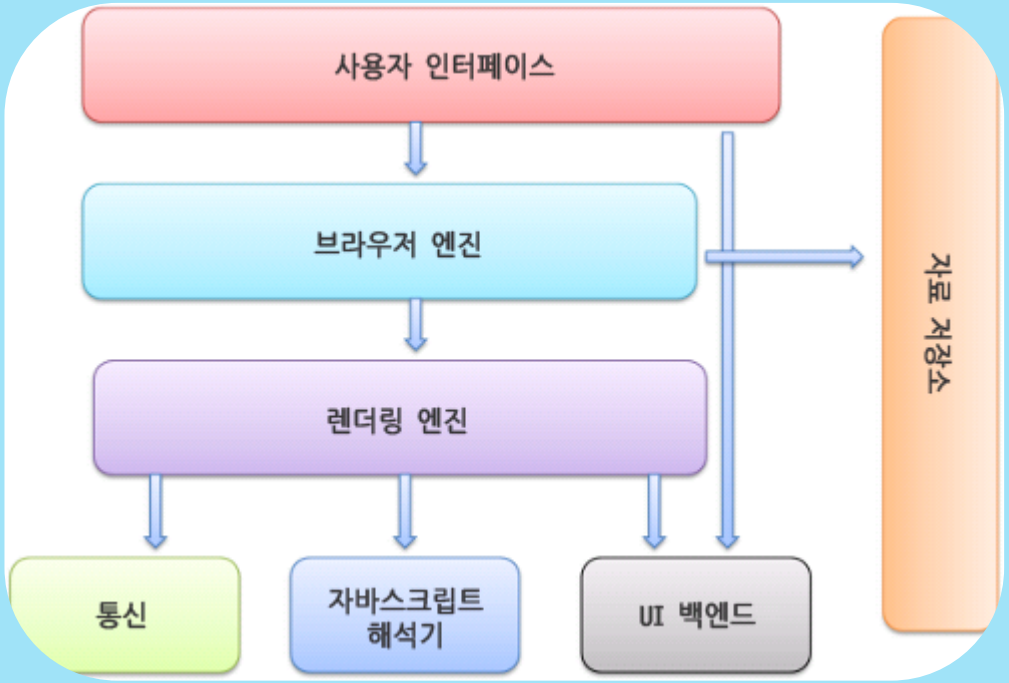
사용자 인터페이스와 렌더링 엔진 사이 동작 제어

3) 렌더링 엔진

요청한 콘텐츠를 표시(HTML 요청하면 HTML, CSS를
파싱해 화면에 표시)

참고 : https://d2.naver.com/helloworld/59361

브라우저의 기본 구조(2)



4) 통신

HTTP 요청과 같은 네트워크 호출에 사용됨
(플랫폼 독립적인 인터페이스, 각 플랫폼 하부에서 실행됨)

5) UI 백엔드

콤보 박스와 창 같은 기본적인 장치를 그림
플랫폼에서 명시하지 않은 일반적인 인터페이스
OS 사용자 인터페이스 체계를 사용

6) 자바스크립트 해석기

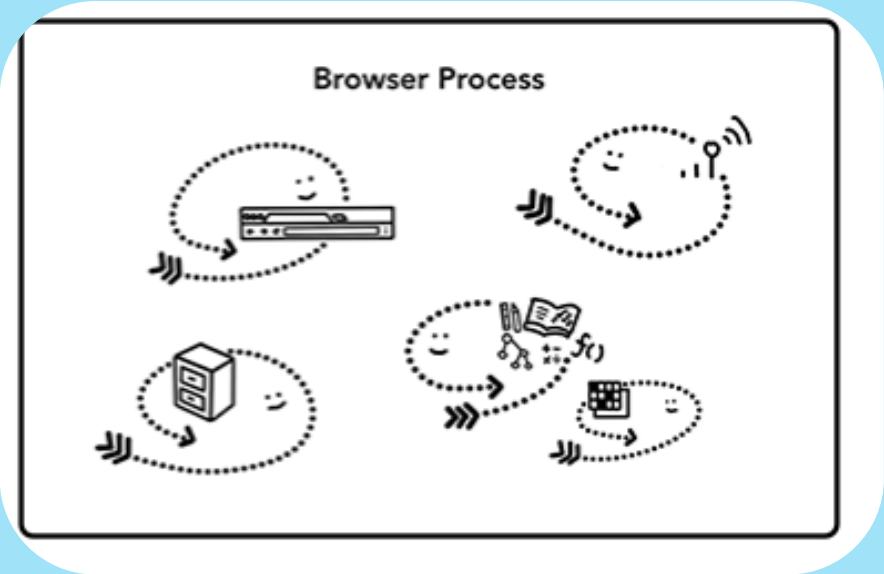
자바스크립트 코드를 해석하고 실행

7) 자료 저장소

자료를 저장하는 계층
모든 종류의 자원을 하드 디스크에 저장할 필요 있음(예 - 쿠키 저장하는 것)
HTML5 명세에는 브라우저가 지원하는 Web DB가 정의됨

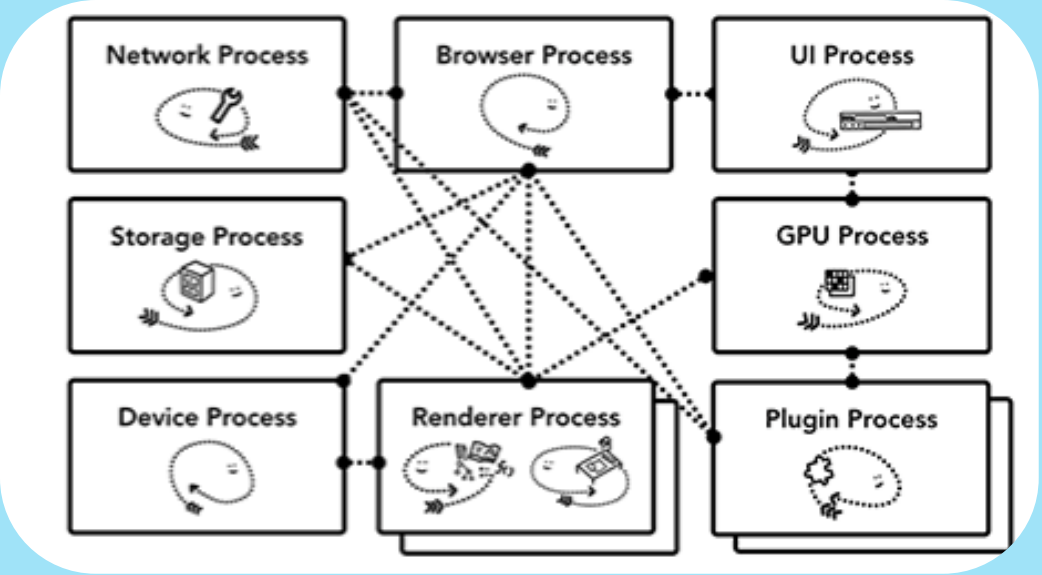
참고 : https://d2.naver.com/helloworld/59361

브라우저 아키텍처 내부 프로세스



웹 브라우저가 어떻게 작동해야 한다는 표준은 없음

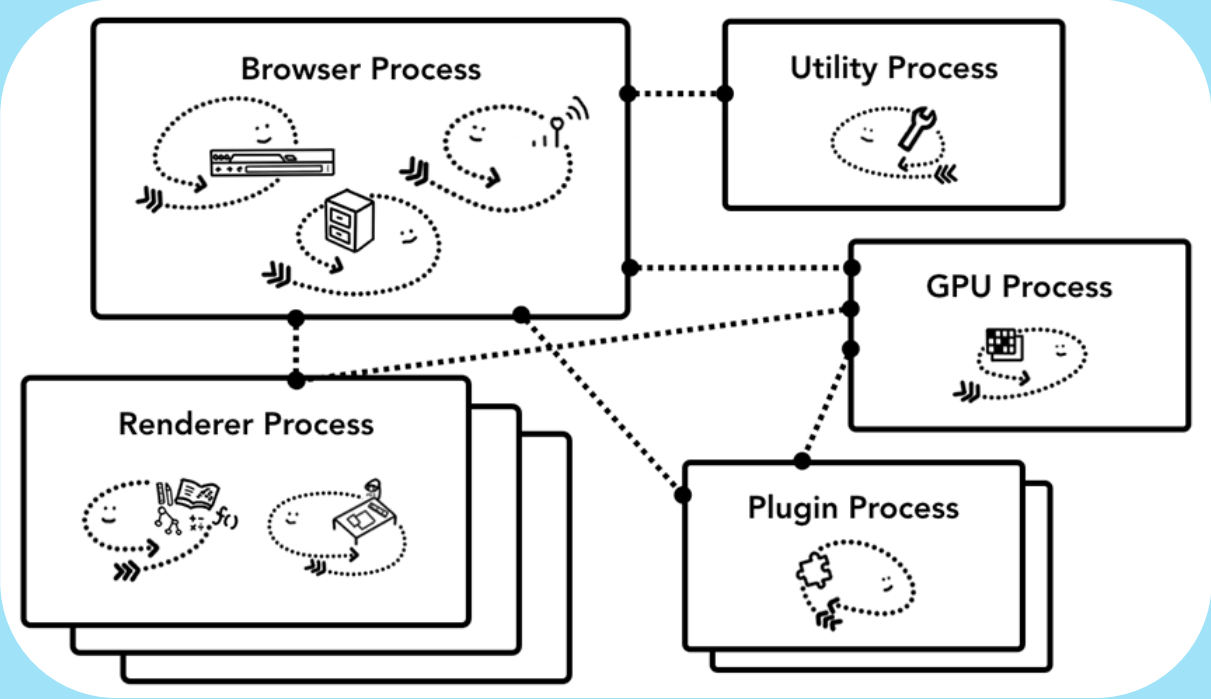
한 브라우저의 접근 방식이 다른 것들과 완전히 다를 수도 있음



크롬의 최근 아키텍처를 기반으로 설명(예정)

참고 : https://developers.google.com/web/updates/2018/09/inside-browser-part1?hl=ko

브라우저 아키텍처 내부 프로세스



브라우저 프로세스(Browser Process)
최상위에서 어플리케이션의 다른 부분을 담당하는 프로세스들을 조율

렌더 프로세스(Render Process)
다수의 프로세스가 생성되어 각 탭마다 할당됨

크롬은 가능하면 각 탭마다 별도의 프로세스를 할당

iframe을 포함해 각 사이트 별로 프로세스를 가지도록 변경
(사이트 격리)

참고 : https://developers.google.com/web/updates/2018/09/inside-browser-part1?hl=ko

브라우저 아키텍처 내부 프로세스

Browser 프로세스

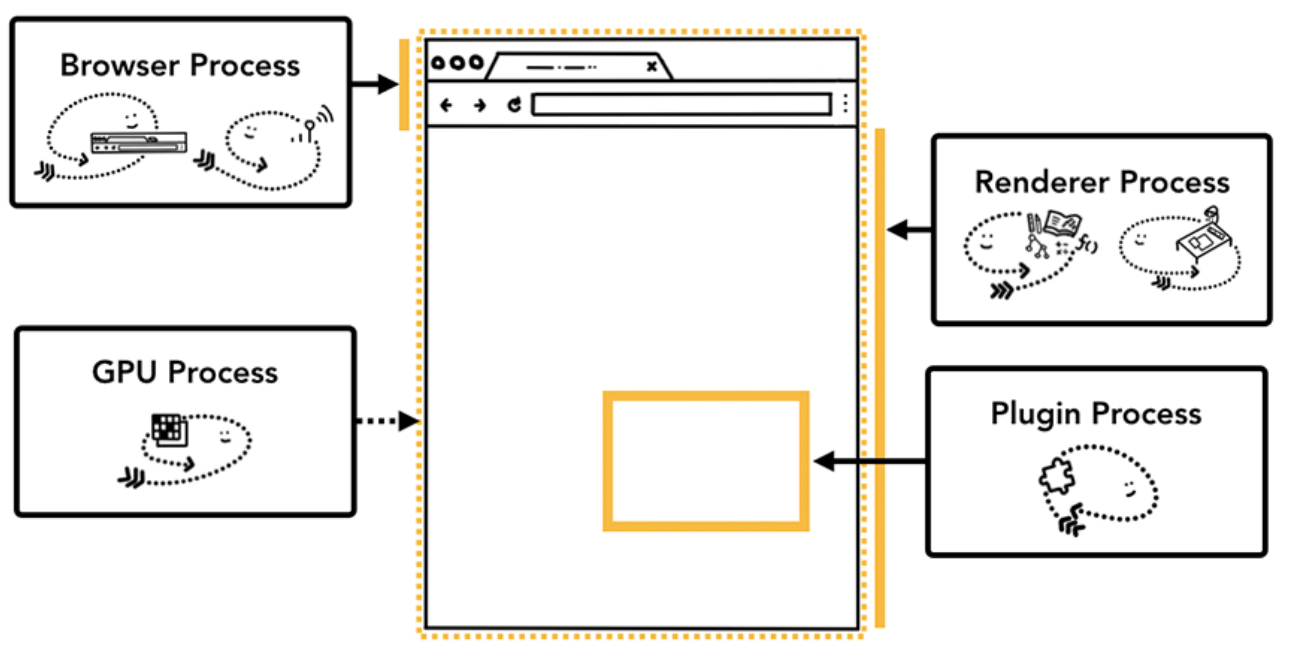
주소 창, 뒤로 및 앞으로 이동 버튼을 포함한 어플리케이션의 “chrome” 부분을 제어

네트워크 요청 및 파일 액세스와 같은 웹 브라우저의 권한이 부여된 보이지 않는 부분 제어

GPU 프로세스

다른 프로세스와 분리된 GPU 작업을 제어

GPU는 여러 앱의 요청을 제어하고 동일한 표면에 표시하기 때문에 다른 프로세스로 분리됨



Renderer 프로세스

웹 사이트가 디스플레이 될 때 탭 안의 모든 것 담당

Plugin 프로세스

플래시와 같은 웹사이트가 사용하는 모든 플러그인 담당

참고 : https://developers.google.com/web/updates/2018/09/inside-browser-part1?hl=ko

DNS resolution (DNS Lookup)

각 웹 서버(실제로 인터넷에 연결된 모든 호스트)에는 텍스트 형식의 고유한 IP 주소가 있음

텍스트 형식 => IP 주소(207.142.131.248)로 변환하는 과정

1. 변환된 IP 주소를 반환하는 DNS 서버에 접속

변환 과정은 하나의 DNS 서버에서만 발생하지 않을 수도 있습니다.

처음에 접속한 DNS 서버는 번역을 완료하기 위해 다른 DNS 서버를 재귀적으로 호출 가능

참고 : https://nlp.stanford.edu/IR-book/html/htmledition/dns-resolution-1.html

DNS resolution : 웹 크롤링에서 잘 알려진 병목 현상

DNS의 분산 특성으로 인해

DNS 확인에는 인터넷을 통한 여러 요청,왕복이 수반될 수 있음

몇 초, 때로는 더 오래 걸릴 수도 있음

해결책 : 캐싱을 도입하는 것

최근 DNS 조회를 수행한 URL은 DNS 캐시에서 찾을 수 있음

참고 : https://nlp.stanford.edu/IR-book/html/htmledition/dns-resolution-1.html

DNS - Domain Name System

사람은 도메인 이름을 통해 온라인으로 정보에 접근 But 웹 브라우저는 IP 주소를 통해 상호작용을 함

브라우저가 인터넷 자원을 로드할 수 있게 도메인 주소 ---〉 IP로 변환

why? 인터넷에 연결된 기기에는 다른 컴퓨터가 기기를 찾는 데 사용하는 고유한 IP주소가 존재

DNS 서버를 사용하면 사람이 영문과 숫자로 된 복잡한 IP주소를 기억할 필요가 없음

참고 : https://www.cloudflare.com/ko-kr/learning/dns/what-is-a-dns-server/

DNS 서버의 종류

1. DNS Recursive Resolver

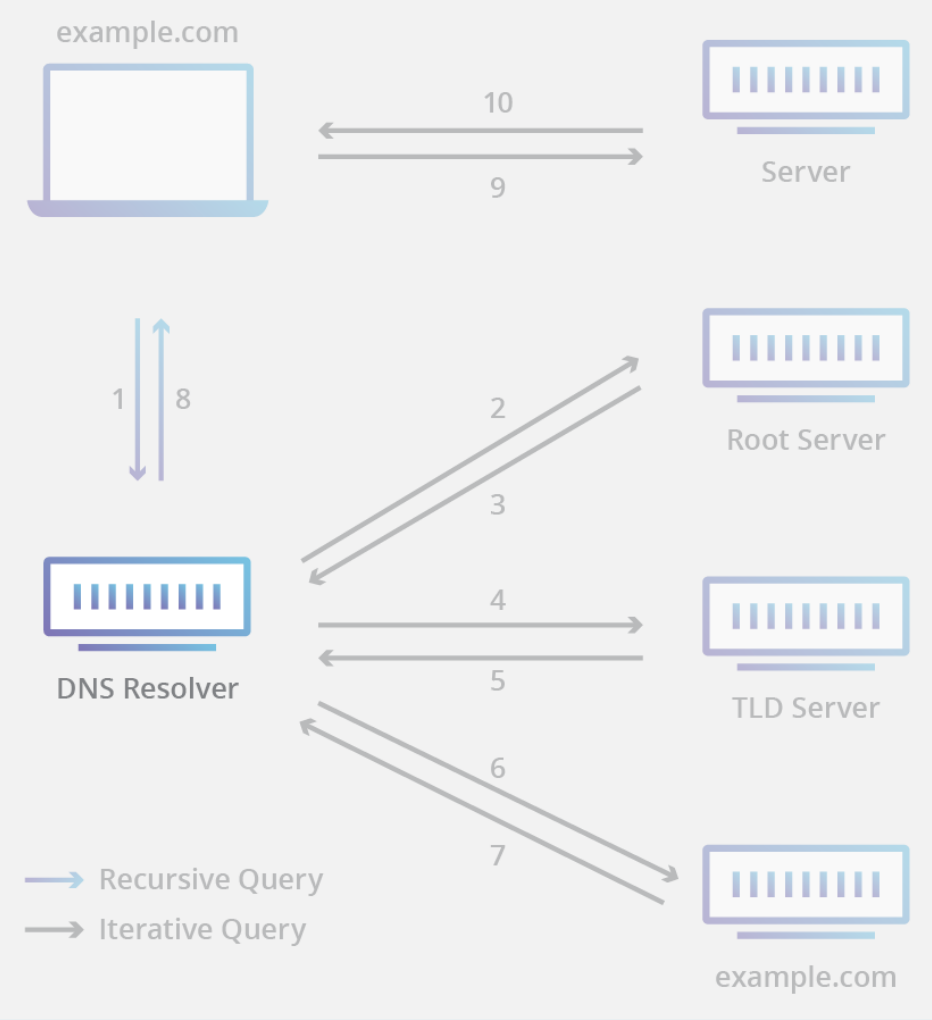
2. Root Name Server

3. TLD Name Server

4. Authenticated Name Server

참고 : https://www.cloudflare.com/ko-kr/learning/dns/what-is-a-dns-server/

DNS recursive resolver(DNS recursor)



DNS 쿼리의 첫 단계, 클라이언트와 DNS 네임서버 사이의 중개자 역할

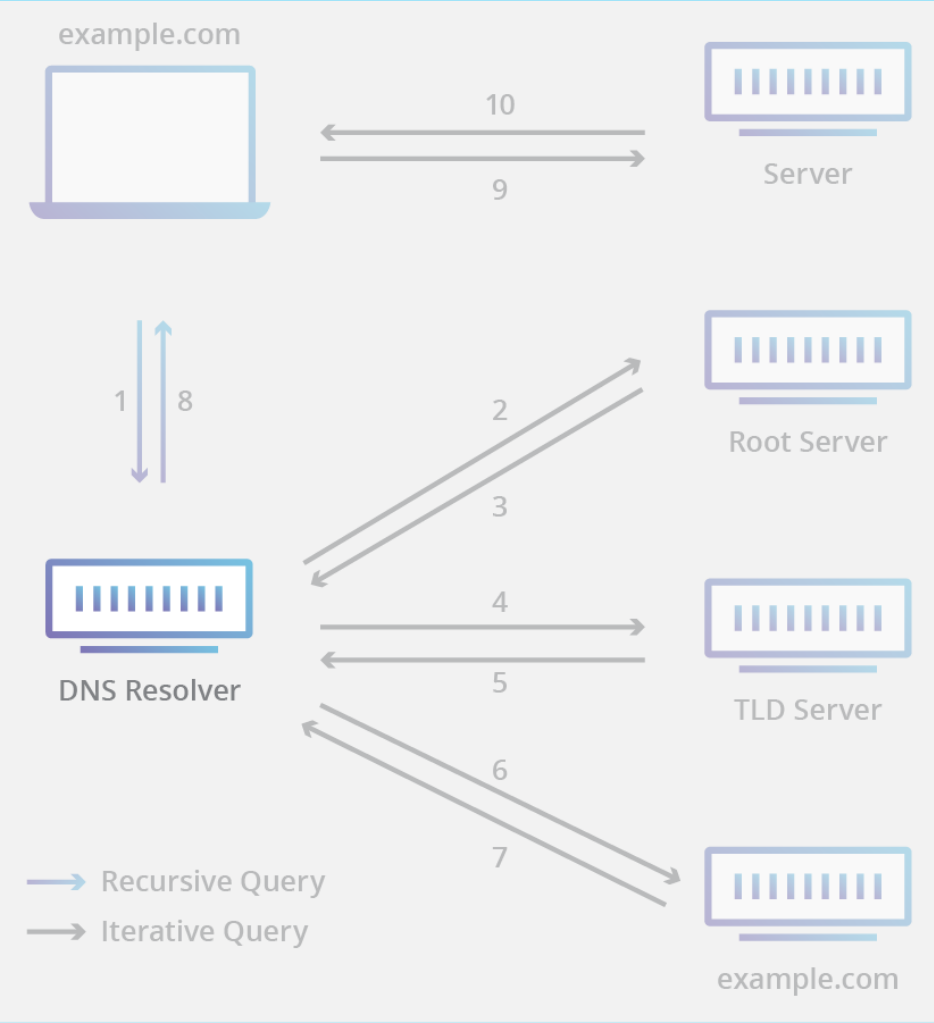
웹 클라이언트로부터 DNS 쿼리를 받은 후 2가지 경우,

1) 캐시된 데이터로 응답

2) 요청을 Root Name Server로 보내고,
또 다른 요청을 TLD Name Server로 보낸 후,
마지막 요청을 Authenticated NameServer로 보냄

참고 : https://www.cloudflare.com/ko-kr/learning/dns/what-is-a-dns-server/

DNS recursive resolver(DNS recursor)



요청된 IP 주소가 있는 Authenticated Name Server로부터
응답을 받은 후 응답을 클라이언트에 보냄

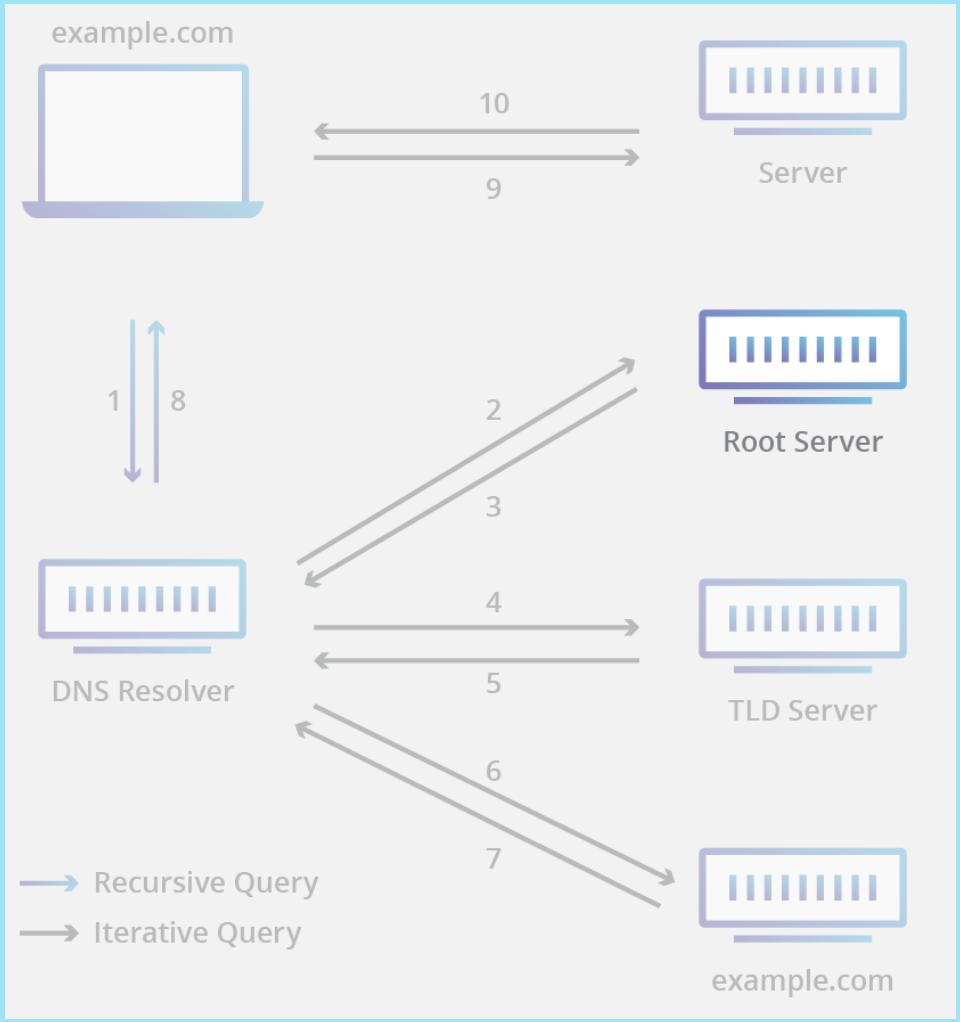
과정 중, DNS recursive resolver는
Authenticated Name Server에서 받은 정보를 캐시

클라이언트가 다른 클라이언트가 최근에 요청한
도메인 이름의 IP 주소를 요청하면,

DNS recursive resolver가 Nameserver와의 통신 Process를
우회하고 캐시에서 요청한 레코드를 클라이언트에 전달 가능

참고 : https://www.cloudflare.com/ko-kr/learning/dns/what-is-a-dns-server/

DNS Root NameServer(DNS recursor)



DNS recursive resolver가 DNS 레코드를 요청하는 과정의 첫 단계

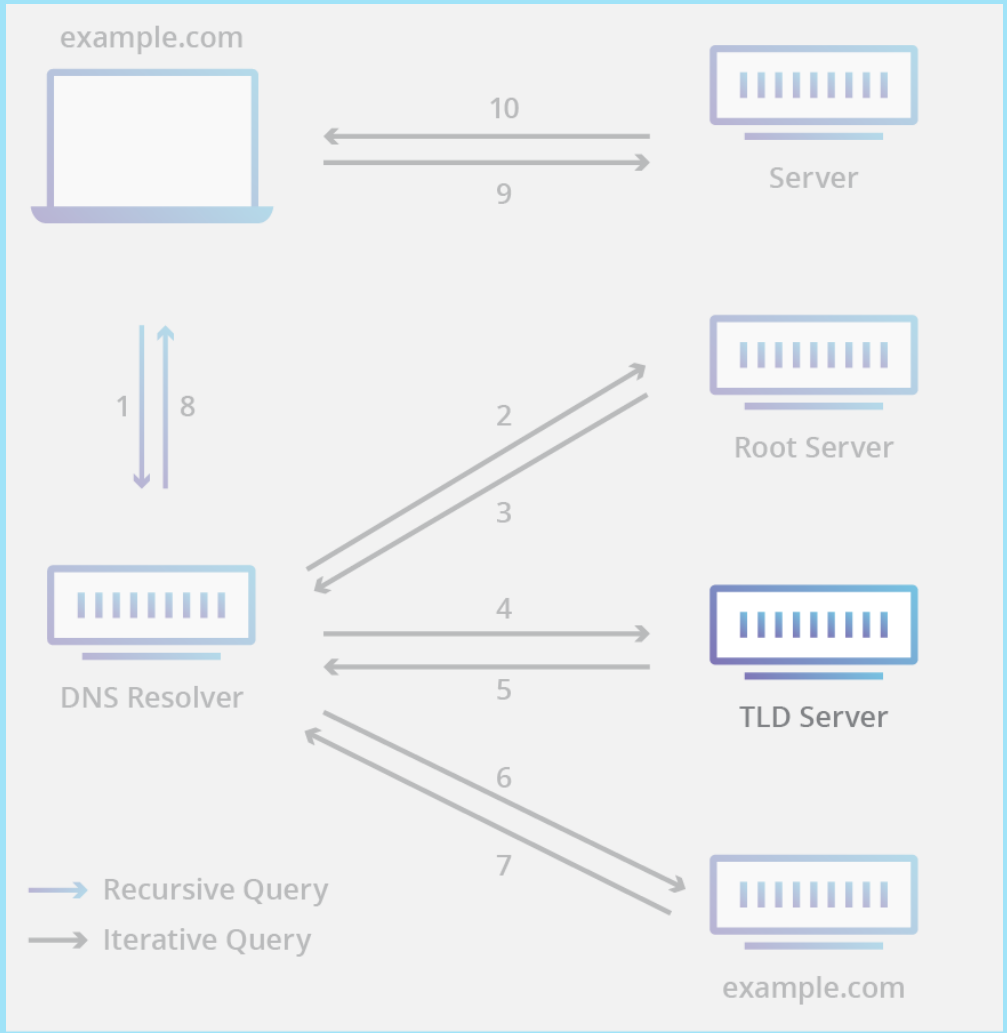
Root Server는 DNS recursive resolver의 쿼리를 수용하며,

Root NameServer는 해당 도메인의 확장자(.com, .net. org 등)에 따라 DNS recursive resolver를 TLD 네임서버에 보내 응답

Root Server는 ICANN이 관리

참고 : https://www.cloudflare.com/ko-kr/learning/dns/dns-server-types/

TLD NameServer (Top-Level Domain)



.com, .net 또는 URL의 마지막 점 뒤에 오는 것 같은
일반적인 도메인 확장자를 공유하는 모든 도메인 이름의 정보를 유지

ex) TLD NameServer - .com으로 끝나는 모든 웹사이트의 정보를 가짐

❶ 사용자가 google.com을 검색하는 경우

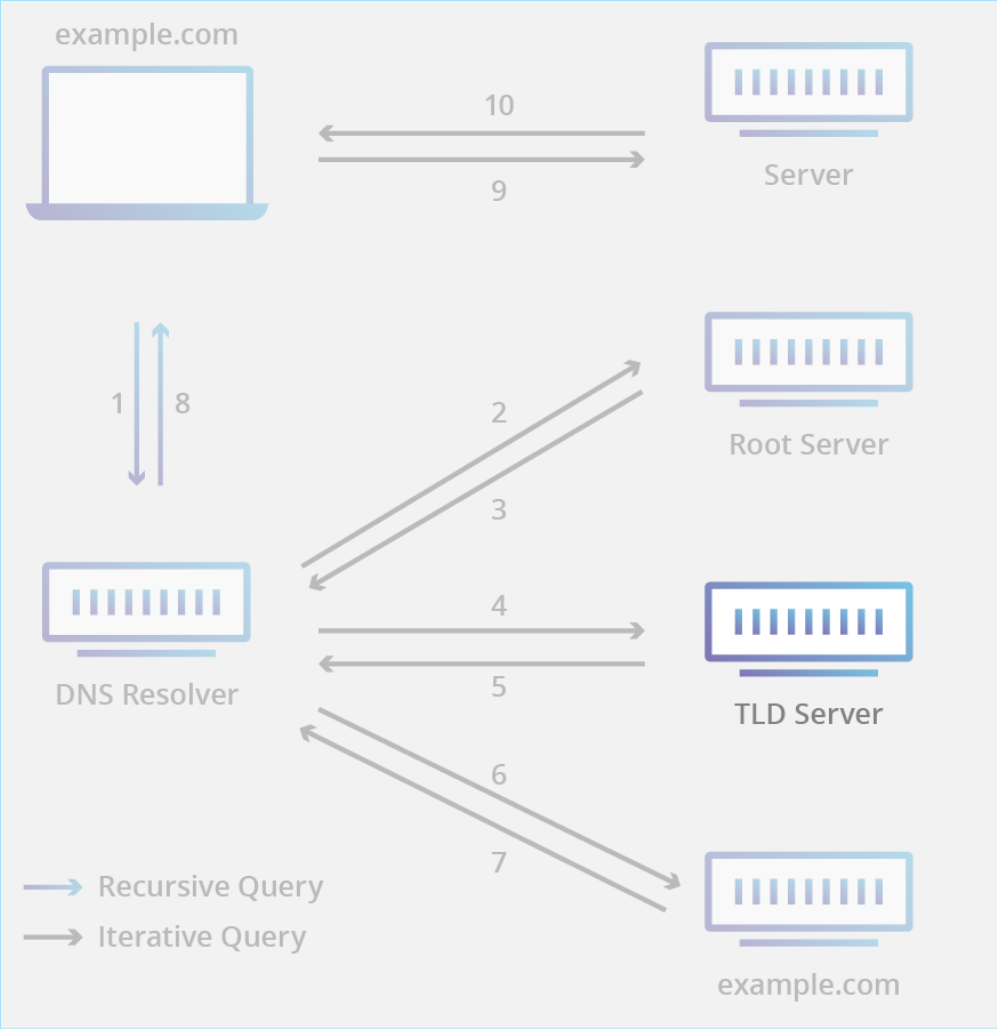
❷ DNS Resolver는 Root NameServer로부터 응답을 받은 후,

❸ 쿼리를 .com TLD NameServer에 보내고

❹ 해당 NameServer는 해당 도메인의
Authenticated NameServer를 가리켜 응답함

참고 : https://www.cloudflare.com/ko-kr/learning/dns/dns-server-types/

TLD NameServer (Top-Level Domain)



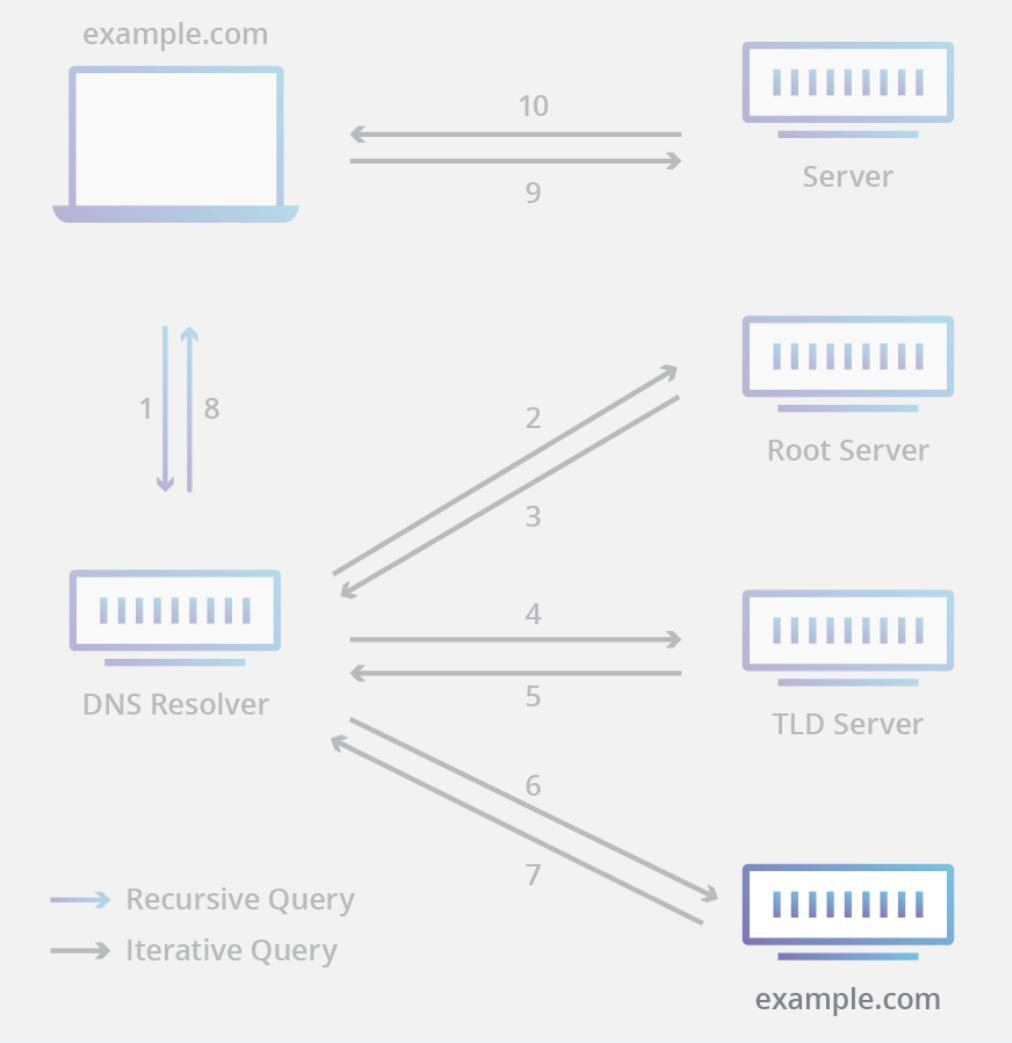
ICANN의 지사인 IANA가 관리(IANA는 TLD 서버를 두 가지로 구분)

1) 일반 최상위 도메인 : 국가별로 고유하지 않은 도메인
(.com, .org, .net, .edu, .gov)

2) 국가 코드 최상위 도메인 : 국가 또는 주와 관련된 모든 도메인
(.uk, .us, .ru, .jp)

참고 : https://www.cloudflare.com/ko-kr/learning/dns/dns-server-types/

Authoritative NameServer



DNS Resolver가 TLD NameServer로부터 응답을 받으면,

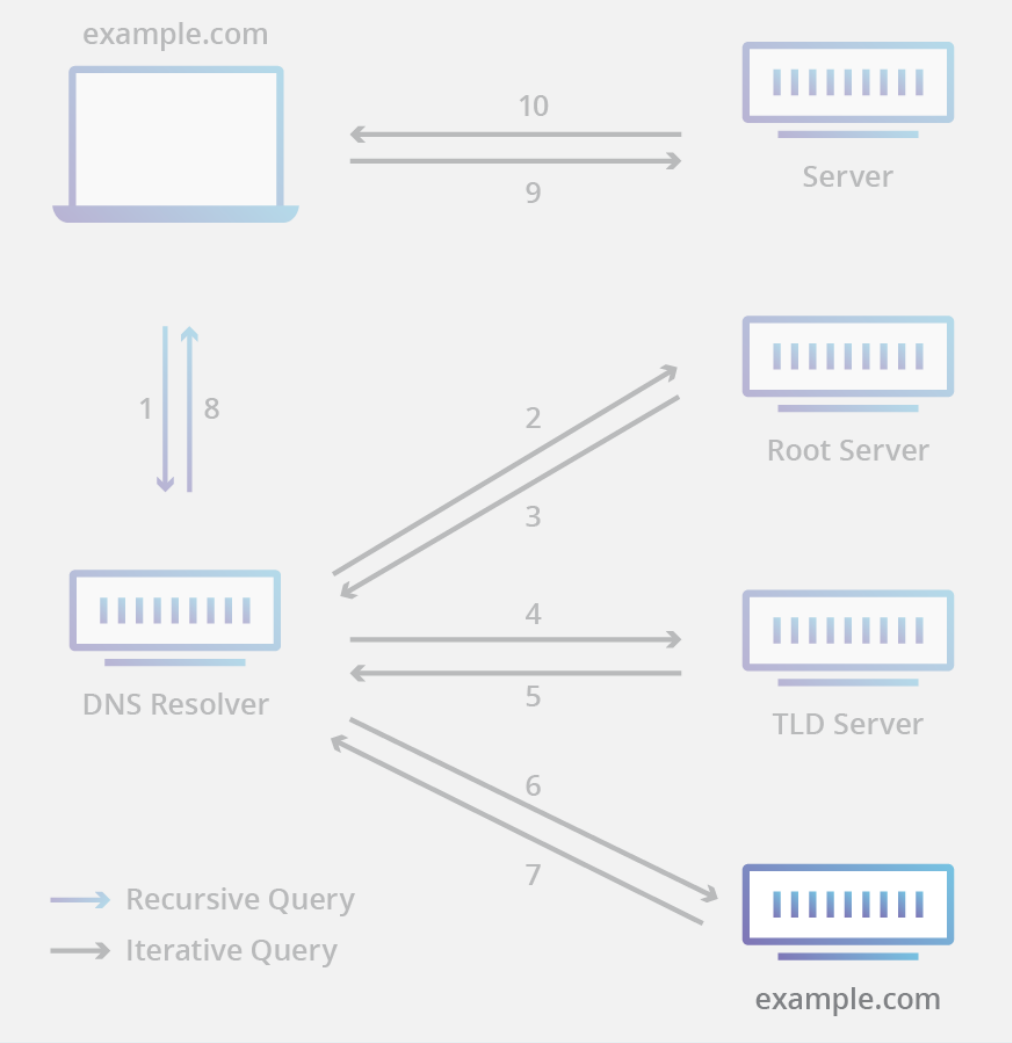
DNS Resolver는 해당 응답을 Authoritative NameServer로 보냄

IP주소를 확인하는 DNS Resolver의 마지막 단계

도메인 이름에 고유한 정보(ex - google.com)를 포함해 DNS A record에서 찾은 도메인의 IP 주소를 DNS Resolver에 제공

참고 : https://www.cloudflare.com/ko-kr/learning/dns/dns-server-types/

Authoritative NameServer



도메인에 CNAME 레코드가 있는 경우

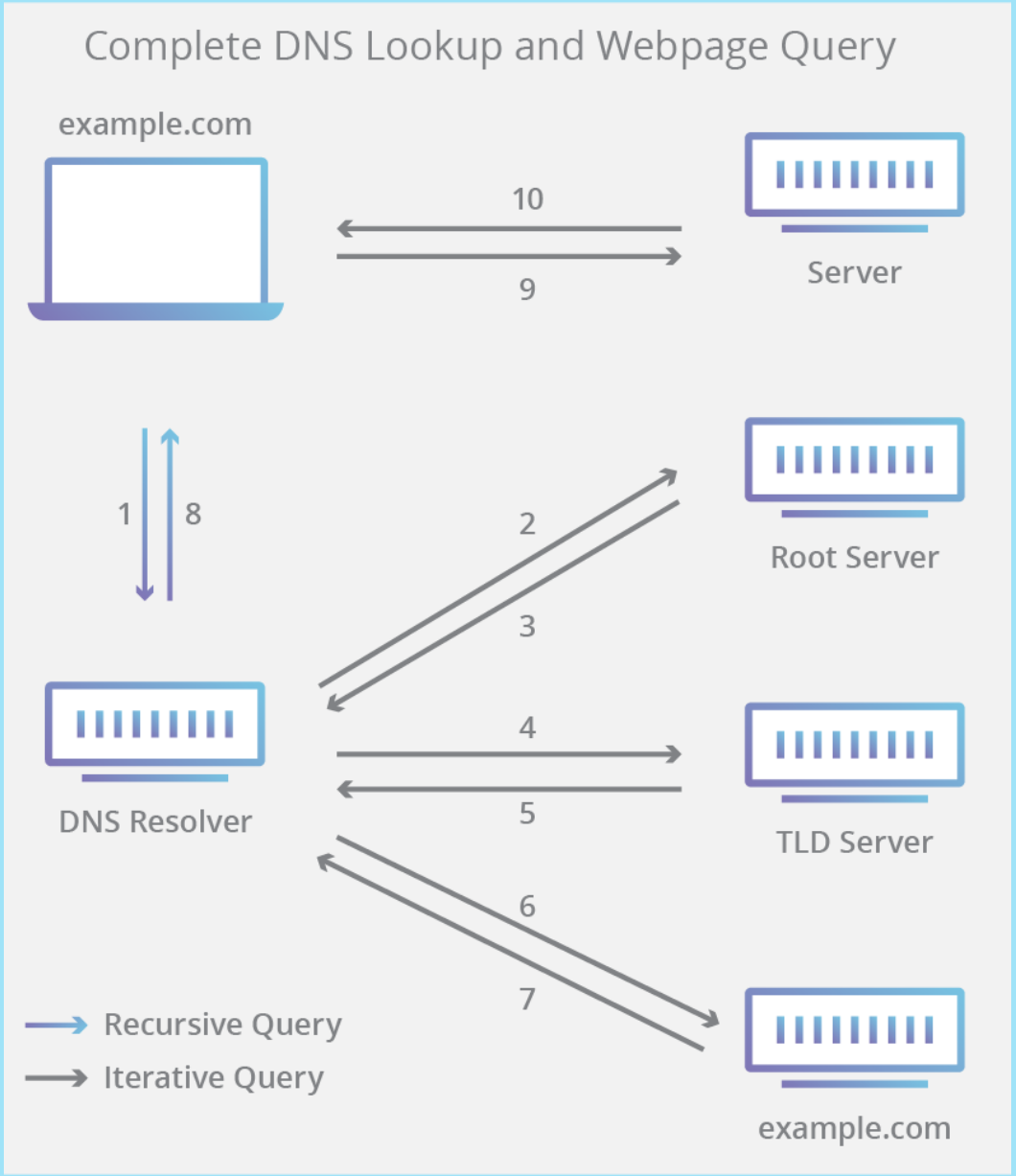
DNS Resolver에 별칭 도메인을 제공

DNS Resolver는 Authoritative NameServer에서

레코드(ex-IP주소를 포함하는 A 레코드)를 얻기 위해

완전히 새로운 DNS 조회를 수행해야 함

참고 : https://www.cloudflare.com/ko-kr/learning/dns/dns-server-types/



DNS Lookup

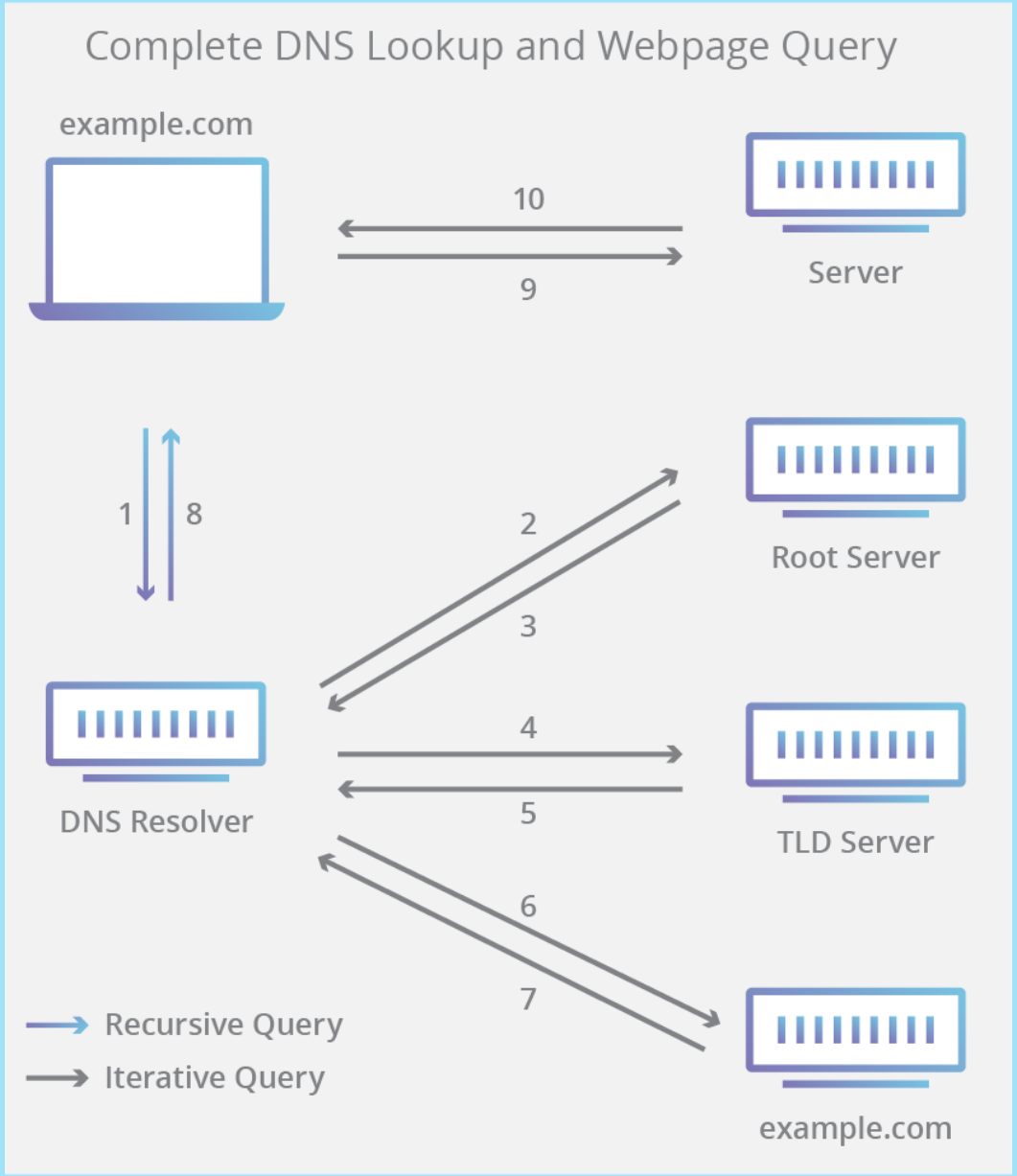
DNS 조회 정보는 쿼리 컴퓨터 내부에서 로컬로 or
DNS 인프라에서 원격으로 캐시되는 경우가 많음

DNS 조회는 일반적으로 8단계가 있음,

DNS 정보가 캐시되어 있으면
DNS 조회 단계에서 몇 단계를 건너뛸 수 있음

=> 더 빨라진다는 의미!

참고 : https://www.cloudflare.com/ko-kr/learning/dns/dns-server-types/



DNS Lookup

1. 사용자가 웹 브라우저에 example 입력

쿼리가 인터넷으로 이동하고 DNS Resolver가 이를 수신

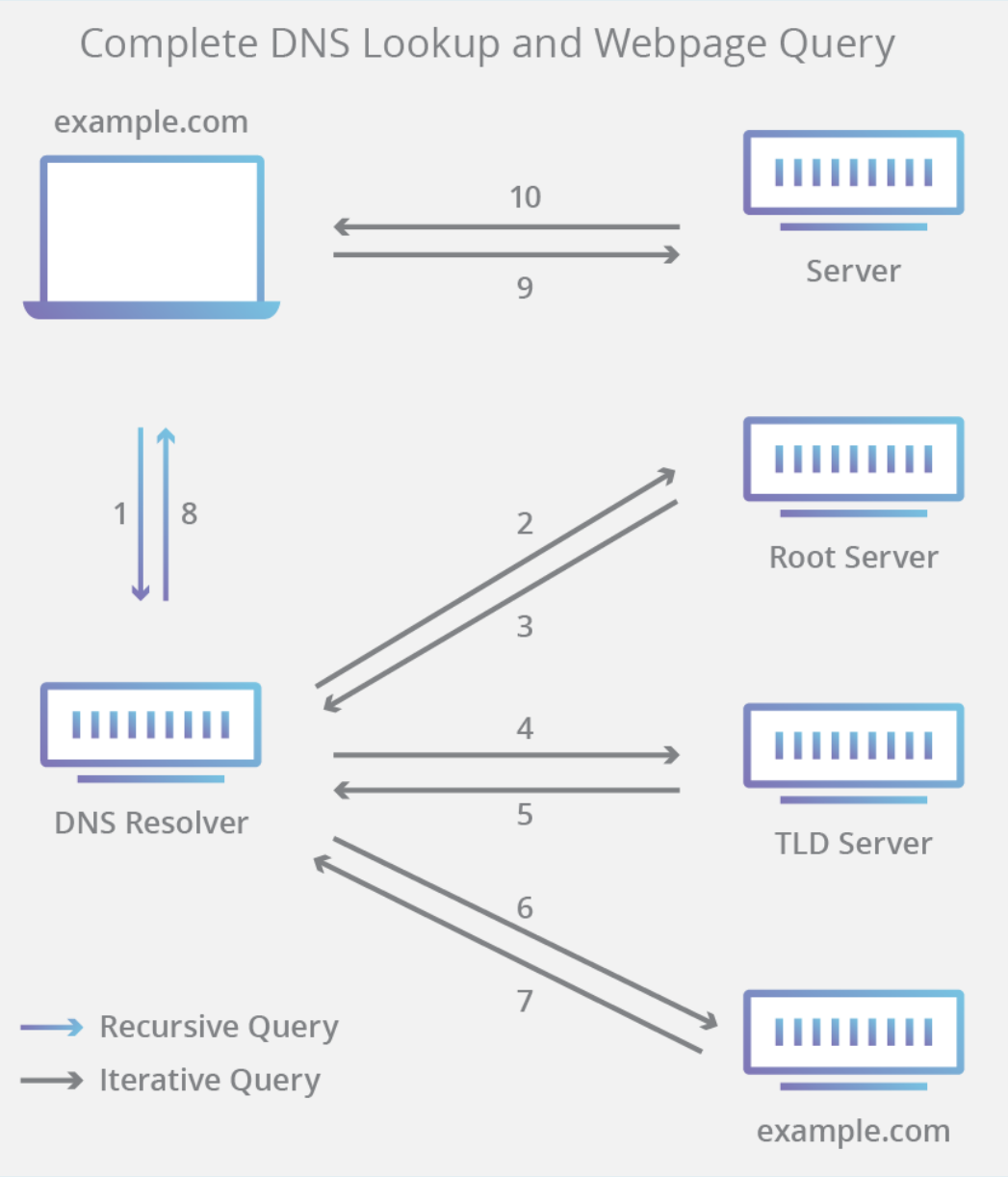
2. DNS Resolver가 DNS Root NameServer를 쿼리

3. Root Server가 도메인에 대한 정보를 저장하는 최상위 도메인(TLD) DNS 서버(.com, .net)의 주소로 DNS Resolver에 응답

4. DNS Resolver가 .com TLD에 응답

5. TLD 서버가 도메인 이름 서버(example.com)의 IP주소로 응답함

참고 : https://www.cloudflare.com/ko-kr/learning/dns/what-is-dns/?__cf_chl_captcha_tk__=d2qcvjJ1HS1zS9Ruizl17UkhQLGKJpQdOHqPBDSJH_Q-1641885216-0-gaNycGzNCr0



DNS Lookup

6. 마지막으로 DNS Resolver가 DNS NameServer로 쿼리를 보냄

7. example.com의 IP 주소가 NameServer에서 DNS Resolver로 반환됨

8. DNS Resolver가 처음 요청한 도메인의 IP 주소로 웹 브라우저에 응답

--- example.com의 IP주소가 반환되면, 브라우저가 웹 페이지를 요청할 수 있음

9. 브라우저가 IP 주소로 HTTP 요청을 보냄

10. 해당 IP의 서버가 브라우저에서 렌더링할 웹 페이지를 반환

참고 : https://www.cloudflare.com/ko-kr/learning/dns/what-is-dns/?__cf_chl_captcha_tk__=d2qcvjJ1HS1zS9Ruizl17UkhQLGKJpQdOHqPBDSJH_Q-1641885216-0-gaNycGzNCr0

- HTTP exchange -

브라우저가 요청을 처리할 서버를 식별

해당 서버와 TCP 연결을 시작하고 HTTP 교환을 시작

HTTP? 웹에서 통신하는 데 가장 널리 사용되는 프로토콜 이름

HTTP 교환은 클라이언트가 요청을 보내고 서버가 응답으
로 응답하는 것을 포함

브라우저가 google.com 뒤 서버에 성공적으로 연결된 후,
요청을 보냄

HTTP exchange - By Client

```
GET / HTTP/1.1
Host: google.com Accept: */*
```

① GET / HTTP/1.1
브라우저는 서버의 / 위치에서 문서를 검색하도록 요청하고 나머지 요청은 HTTP/1.1 프로토콜을 따름

② Host: google.com
HTTP/1.1에서 필수인 유일한 HTTP 헤더,
서버가 여러 도메인(google.com, google.co.uk 등)을 제공할 수 있기 때문에
클라이언트는 요청이 해당 특정 호스트에 대한 것이라고 알 수 있음

③ Accept: */*
브라우저가 서버에 모든 종류의 응답을 다시 수락할 것이라고 알리는 선택적 헤더
서버는 JSON, XML, HTML 형식으로 사용할 수 있는 리소스를 가질 수 있으므로 선호하는 형식을 선택 가능

참고 : https://www.freecodecamp.org/news/web-application-security-understanding-the-browser-5305ed2f1dac/

HTTP exchange - By Server

```
HTTP/1.1 200 OK
Cache-Control: private,
max-age=0
Content-Type: text/html;
charset=ISO-8859-1
Server: gws
X-XSS-Protection: 1;
mode=block
X-Frame-Options: SAMEORIGIN
Set-Cookie: NID=1234;
expires=Fri, 18-Jan-2019 18:25:04 GMT;
path=/; domain=.google.com; HttpOnly
```

클라이언트 역할을 하는 브라우저가 요청을 마치면 서버가 응답할 차례

요청이 성공했고, 몇가지 헤더를 추가
- 요청을 처리한 서버, X-XSS-Protection 정책이 무엇인지 알려주는 등

클라이언트와 서버가 HTTP를 통해 정보를 교환

참고 : https://www.freecodecamp.org/news/web-application-security-understanding-the-browser-5305ed2f1dac/

Rendering

클라이언트와 서버가 HTTP를 통해 정보를 교환

응답 본문에서 서버는 Content-Type 헤더에 따른 응답 표현을 포함

text/html로 설정되어 응답에서 HTML 마크업이 예상되는 것을 알 수 있음

HTML을 구문 분석하고 마크업에 포함된 추가 리소스(가져올 JS, CSS 문서 등)을 로드하고
가능한 한 빠르게 사용자에게 제공

참고 : https://www.freecodecamp.org/news/web-application-security-understanding-the-browser-5305ed2f1dac/

Thank You!

39ghwjd@naver.com