

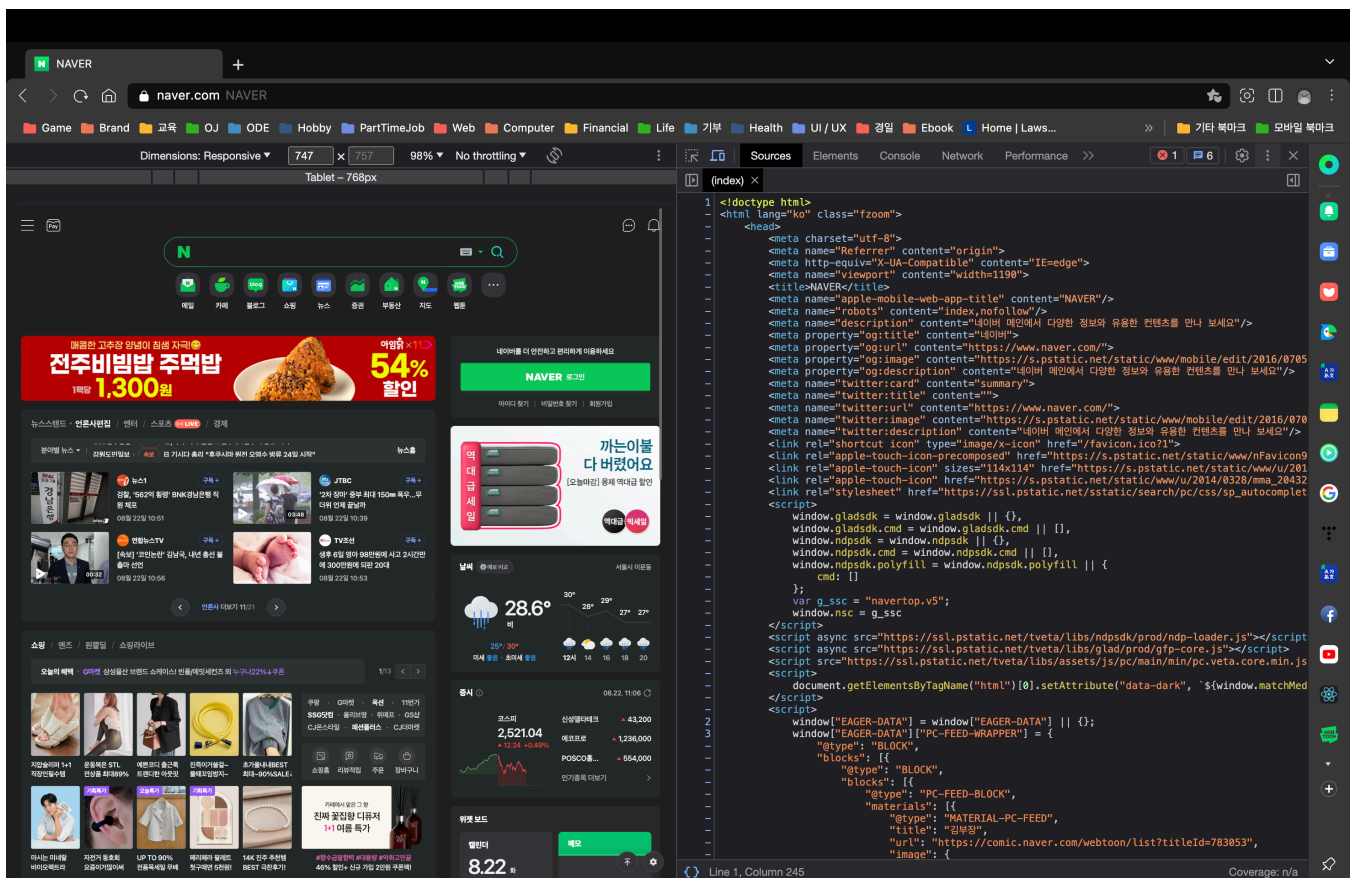
1. 웹 브라우저가 메시지를 만든다.

네이버에 접속하기 위해 웹 브라우저의 주소창에 `www.naver.com` 이라고 입력했을 때, 가장 먼저 웹 브라우저가 하는 일은 메시지를 만드는 것이다. 이번 챕터에서는 어떤 과정을 거쳐 메시지가 작성되는지 알아보도록 하자.

1-1. HTTP 리퀘스트 메시지를 작성한다.

URI

주소창에 작성했던 `www.naver.com` 와 같은 것을 URL(Uniform Resource Locator)이라고 한다. 사실 우리가 웹 페이지(Web Page)라고 부르는 것의 실체는 HTML(Hypertext Markup Language) 문서로 웹 브라우저가 이를 분석해 우리에게 보여주는 것이다. 다시 말해 `www.naver.com` 를 입력하면 네이버의 서버에 어떤 HTML 문서 파일을 요청하는 것이라고 할 수 있다.



The screenshot shows a web browser with the Naver homepage loaded. The address bar shows `naver.com`. The browser's developer tools are open, displaying the source code of the page. The source code is an HTML document with a `<doctype html>` declaration and a `<html>` tag. It includes meta tags for charset, referrer, viewport, and title. The title is "NAVER". The source code also includes a `<script>` tag for a JavaScript library, `jquery.js`, and a `<script>` tag for a custom script, `jquery.js`. The source code is displayed in a dark theme.

우리가 보는 네이버는 오른쪽의 HTML 문서를 분석한 결과다.

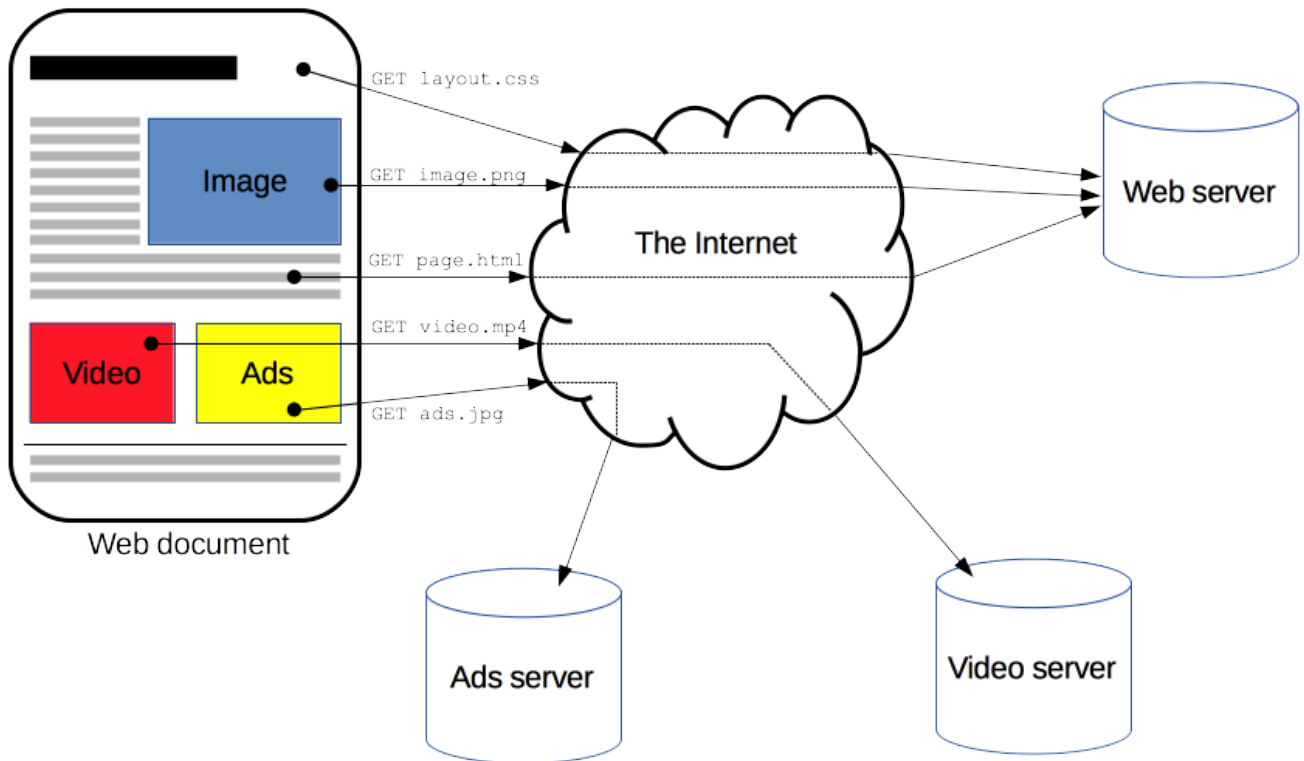
서버에는 여러 가지 자원이 있기 때문에 우리가 원하는 자원을 요청하려면 이를 식별할 수 있어야 한다. 웹에서는 이를 위해 URI(Uniform Resource Identifier)를 사용한다. URI는 URL과 URN(Uniform Resource Name)으로 나뉘는데, URL은 자원의 위치를 나타내고 URN은 자원의 이름을 나타낸다.* URI 문법에 대한 설명은 [여기](#)로 대체한다.

* URN보다는 URL을 더 많이 사용하긴 한다.

HTTP

HTTP(Hypertext Transfer Protocol)는 이름에서도 알 수 있듯 HTML 문서를 전송하기 위한 프로토콜*이다. 하지만 지금은 HTML 문서 외에도 이미지, 비디오 등 다양한 자료를 전송할 수 있게 됐다.

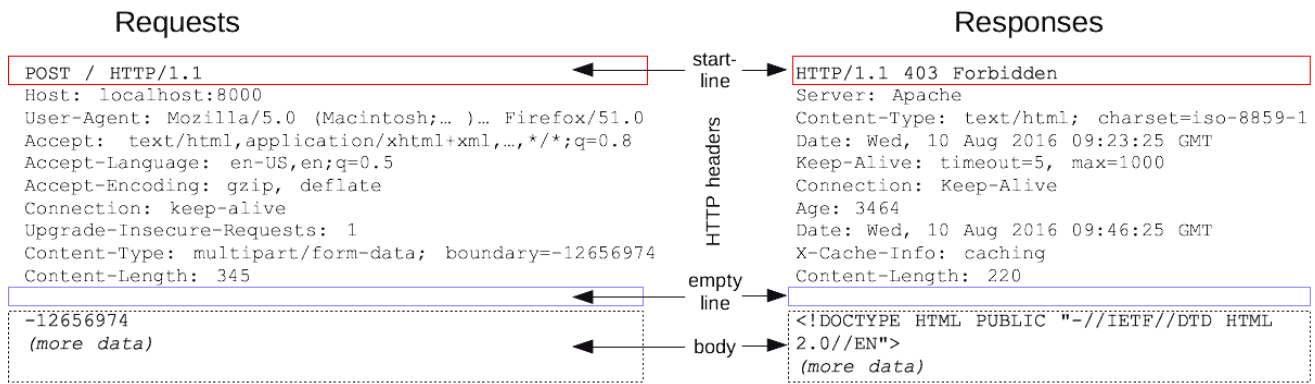
* 프로토콜은 통신을 위한 규칙을 말한다.



하나의 웹 문서를 구성하기 위해 여러 번의 HTTP 호출이 필요한 경우도 있다.

HTTP의 동작은 서버에 무언가 요청하면 그에 대한 응답을 받는 것으로 구성된다. 요청 서식과 응답 서식이 정해져 있는데 이를 각각 요청 메시지*(Request Message)와 응답 메시지(Response Message)라고 한다.

* 메시지란 통신되는 데이터를 지칭한다.



요청 메시지와 응답 메시지의 예시

메시지를 잘 들여다보면 구성이 크게 다르지 않은 것을 알 수 있다. 시작 줄에는 요청 혹은 응답의 내용을 적고, 여러 가지 부가 정보가 담긴 헤더*(Header)를 작성한다. 그 후 한 줄의 개행 후에 바디(Body)를 작성하는데, 여기에는 요청과 관련된 데이터나 요청에 대한 결과가 포함된다.

* 헤더의 각 항목에 대해서 알고 싶다면 [여기](#)를 참고하라.

메소드

요청 메시지의 시작 줄에는 메소드(Method)가 포함된다. 메소드는 메소드는 HTTP의 동작 방법을 의미하며 종류는 아래와 같다. 각각에 대해서 자세히 알고 싶다면 [여기](#)를 참고하라.

메소드	설명
GET	데이터를 요청한다.
HEAD	데이터에 대한 정보를 요청한다.
POST	데이터를 전송한다.
PUT	서버의 파일을 교체한다.
DELETE	서버의 파일을 삭제한다.
CONNECT	서버와의 터널을 구성한다.
OPTIONS	통신 옵션을 통지하거나 조사한다.
TRACE	메시지 루프백 테스트를 수행한다.
PATCH	자원의 수정된 내역을 적용한다.

상태 코드

응답 메시지의 시작 줄에는 상태 코드(Status Code)가 포함된다. 상태 코드는 요청이 성공적으로 완료되었는지를 나타낸다. 상태 코드는 5개의 그룹으로 묶여있다. 각각에 대해서 자세히 알고 싶다면 [여기](#)를 참고하라.

상태 코드 값	설명
100 ~ 199	처리의 경과 상황 등을 통지한다.
200 ~ 299	요청 성공적으로 완료됐음을 알린다.
300 ~ 399	다른 조치가 필요함을 나타낸다.
400 ~ 499	클라이언트 오류를 나타낸다.
500 ~ 599	서버 오류를 나타낸다.

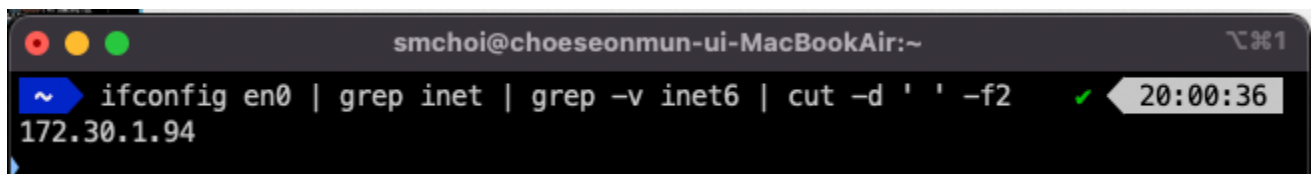
1-2. 웹 서버의 IP 주소를 DNS 서버에 조회한다.

IP 주소

인터넷에는 수많은 컴퓨터가 접속되어 있다. 이중 우리가 원하는 컴퓨터에 HTTP 요청을 하려면 해당 컴퓨터를 찾아낼 수 있는 정보가 필요한데, 여기에는 IP(Internet Protocol) 주소를 사용한다.

IP 주소는 32비트의 숫자이며*, 네트워크 ID와 호스트 ID로 구성 된다. 표기는 `221.149.9.93` 과 같이 8비트씩 끊어서 십진수로 표기하는 십진수 점 표기법(DDN; Dotted Decimal Notation)을 사용한다.

* 엄밀히는 IPv4 주소 길이이며, IPv6 주소의 길이는 128비트이다.



```
smchoi@choeseonmun-ui-MacBookAir:~  
~ ifconfig en0 | grep inet | grep -v inet6 | cut -d ' ' -f2 20:00:36  
172.30.1.94
```

인터넷에 접속한 모든 컴퓨터는 IP 주소를 갖고 있다.

서브넷 마스크

`221.149.9.93` 을 다시 보자. 이 주소에서 네트워크 ID와 호스트 ID가 구분이 되는가? 이를 구분하려면 서브넷 마스크(Subnet Mask)가 필요하다. 서브넷 마스크는 `221.149.9.93/24` 처럼 IP 주소 뒤에 네트워크

ID의 길이를 표기한다*. 혹은 DDN이나 이진수로도 표기할 수 있다.

* 이를 CIDR(Classless Inter-Domain Routing)이라고 한다.

특수한 IP 주소

IP 주소 중 몇 가지 특수한 IP 주소가 있다.

- **네트워크(Network) 주소**

네트워크를 나타내는 주소로 호스트 ID의 모든 비트가 0이다. 예를 들어

221.149.9.93/24 에서는 221.149.9.0 가 네트워크 주소다.

- **브로드캐스트(Broadcast) 주소**

네트워크에 속한 모든 호스트에게 데이터를 전송하기 위한 주소로 호스트 ID의 모든 비트가 1이다. 예를 들어 221.149.9.93/24 에서는 221.149.9.255 가 브로드캐스트 주소다.

- **와일드카드(Wildcard) 주소**

0.0.0.0/0 주소를 말하며, 모든 IP를 지칭한다.

- **루프백(Loopback) 주소**

로컬 머신을 지칭하는 주소다. 127.0.0.0/8을 사용한다. 로컬호스트(Localhost)라고도 한다.

도메인 이름

223.130.200.107 라는 IP 주소가 있다고 해보자. 이게 어느 컴퓨터인지 알 수 있는가? 그러면 www.naver.com 은 어떤가? 한눈에 네이버라는 것을 알 수 있다. 223.130.200.107 은 네이버 서버의 IP 주소다.

이처럼 IP 주소는 외우기가 어렵다. 이를 대신하여 사용하는 것이 바로 도메인 이름(Domain Name)이다. 그래서 웹 브라우저는 URL에 적힌 도메인 이름을 DNS(Domain Name System)에 의해 IP 주소로 해석(Resolution)한 후, HTTP 요청을 한다.

도메인 이름 해석을 위해서는 소켓 라이브러리를 사용해야 한다. BSD 소켓 라이브러리를 사용해

www.naver.com 의 해석을 하는 프로그램은 아래와 같다.

```
// Unix OS에서만 동작한다.
```

```
#include <stdio.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <arpa/inet.h>
```

```
void PrintDomain(const struct addrinfo* addresses);
```

```
int main()
{
    const char* domain = "www.naver.com";
    const char* port = "80"; // HTTP

    struct addrinfo hints = { 0 };
    hints.ai_family = AF_INET;
    hints.ai_socktype = SOCK_STREAM;

    struct addrinfo* result = NULL;
    int errorCode = getaddrinfo(domain, port, &hints, &result);

    if (errorCode != 0)
    {
        printf("getaddrinfo error : %s\n", gai_strerror(errorCode));

        exit(EXIT_FAILURE);
    }

    PrintDomain(result);

    freeaddrinfo(result);

    exit(EXIT_SUCCESS);
}
```

```
void PrintDomain(const struct addrinfo* addresses)
{
    puts("Result");

    while (addresses != NULL)
    {
        struct sockaddr_in* address = (struct sockaddr_in*)addresses->ai_addr;
```

```
printf("%s\n", inet_ntoa((struct in_addr)address->sin_addr));

addresses = addresses->ai_next;
}
}

// Output:
// Result
// 223.130.195.200
// 223.130.195.95
```

참고자료

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Identifying_resources_on_the_Web#syntax_of_uniform_resource_identifiers_uris
- <https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages>
- <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>
- <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>
- <https://youtu.be/Yv7KIbbWSow>
- <https://youtu.be/s5klGnaNFvM>
- <https://youtu.be/K1o6sTZng-4>
- <https://youtu.be/9GSvbn6y9uU>
- <https://youtu.be/3-dlUtXHEr0>
- https://youtu.be/JDh_lzHO_CA
- <http://www.ktword.co.kr/test/view/view.php?nav=2&no=1144&sh=CIDR>
- <https://opentutorials.org/course/3276>
- http://www.ktword.co.kr/test/view/view.php?m_temp1=385&id=433