

가사 피피티 메이커 개발 프로젝트

- 웹(WEB)과 URL, HTTPS를 중심으로 -

30923 정준영

목차

I. 서론	3
II. 웹(Web)	4
1. 인터넷(Internet)과 웹(Web).....	4
2. URL(uniform resource locator)	6
0) OSI 모델.....	6
1) 통신 프로토콜.....	6
2) IP.....	8
3) 도메인 네임	9
4) 쿼리 스트링	11
3. DNS(Domain Name System)	12
4. HTTP 프로토콜.....	15
1) HTTP(Hypertext Transfer Protocol)	15
2) HTTPS(Hypertext Transfer Protocol Secure)	16
3) SSL/TLS	17
5. 적용	20
III. 웹 프로그래밍	21
1. HTML	21
2. CSS.....	22
3. JS	24
1) 비동기 함수	25

IV.	웹 개발	28
1.	프론트엔드(front-end)	28
1)	UI(User Interface)	29
2)	UX(User Experience)	29
2.	백엔드(back-end)	30
V.	적용	32
1.	가사 피피티 메이커	32
1)	캡차(Captcha)	37
2)	도메인	38
3)	파비콘(favicon)	39
4)	광고	40
5)	구글 애널리틱스(Google Analytics)	41
2.	홍보	42
1)	블로그	42
2)	가사 피피티 파인더	43
3)	검색	44
VI.	결론	46

I. 서론

초등학생 때 비효율적이고 귀찮은 것이 싫어서 이러한 문제들을 해결하여 많은 사람들이 '편한 세상'에 살 수 있도록 다양한 프로그램이나 앱, 로봇 등을 개발하는 개발자가 되고싶었고 이를 진로로 정하여 지금까지 꿈을 이루기 위해 노력해 왔다.

그러던 중 어느날 교회에서 예배 중 찬양 시간에 사용되는 가사를 표시해주는 PPT 슬라이드를 제작하고 있는 친구를 보고 옆에서 구경을 하게 되었다. 친구는 악보와 콘티 그리고 음원을 들으면서 직접 가사를 타이핑하고 콘티에 맞게 슬라이드를 배열하는 작업을 하고 있었다. 직접 가사를 타이핑하고, 콘티에 맞게 슬라이드를 배열하고, 중복되는 슬라이드를 찾아다니며 복사&붙여넣기 하고 또 가사를 수정하고, 적절한 길이로 가사를 나누어 가독성 좋게 슬라이드를 제작하는 모습을 보며 나는 이러한 과정이 상당히 비효율적이며 조금 더 편하고 쉽게 슬라이드를 만들 수 없을까 라는 고민을 하게 되었다. 이에 대한 해결책으로 가사를 입력하면 알아서 원하는 디자인으로 슬라이드를 만들어주는 프로그램이 있으면 좋겠다고 생각하였고, 많은 사람들이 언제 어디서나 프로그램 설치 없이 사용할 수 있도록 **웹 사이트**로 개발해보자 라는 계획을 세우고 이를 실행하게 되었다.

본 보고서에서는 위와 같은 동기를 가지고 어떤 계획과 과정을 거쳐 어떠한 결과물을 완성하게 되었는지를 순차적으로 설명하겠다. 이전에 수행한 탐구와 중복되는 내용의 경우 설명을 생략하겠다.

내용의 이해를 위해 <https://github.com/jjy0809/creanpl/blob/main/인터넷 댓글 검열 확장프로그램 개발.pdf> 에서 III¹ 과 IV-4²를 읽어 주길 바란다.³

¹ 본 보고서의 **4.HTTP 프로토콜**과 직접적으로 연관

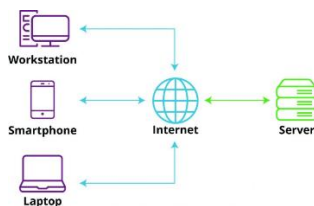
² 본 보고서의 **V.적용**과 직접적으로 연관(해당 부분은 읽지 않으면 생략된 부분이 많아 내용 이해에 큰 어려움이 존재함)

³ 생기부 세특 작성시 위 보고서가 서술될 자율 세특과 내용을 일부 공유하므로 자유롭게 분량 조율 가능

II. 웹(Web)

1. 인터넷(Internet)과 웹(Web)

흔히 말하는 웹은 월드 와이드 웹(World Wide Web, WWW)을 이르는 말로 팀 버너스-리(Tim Berners-Lee)가 1991년 개발한 인터넷 서비스이다. 여기서 인터넷이란 컴퓨터 간의 상호작용을 위한 네트워크 체제로 전세계의 컴퓨터들을 네트워크로 연결시켜주는 역할을 한다. 초기 인터넷은 국방 목적으로 미국에서 1969년 아르파넷(ARPAnet)을 개발하여 폐쇄적으로 사용되었다. 아르파넷은 초기에는 NCP 프로토콜을 사용하였지만 이후 서로 다른 통신망끼리 원활하게 데이터 교환이 가능한 현재도 LAN 통신에 사용하고 있는 TCP/IP 프로토콜로 교체되었고, 이를 계기로 유저가 급증하여 진정한 인터넷의 시대가 열리게 되었다.⁴ 이러한 인터넷 환경에서 유저들에게 HTML 문서를 표시해주는 기술이 필요하였고 해당 기술을 통해 사람들에게 페이지를 표시해주는 서비스가 WWW, 즉 웹이다.



5

인터넷은 기본적으로 클라이언트와 서버간의 통신(client-server model)으로 이루어진다. 클라이언트는 인터넷 네트워크를 통해 서버에 요청을 보내고, 서버는 이를 처리하여 그 결과를 다시 클라이언트에게 응답해주는 방식으로 작동한다. 서버의 종류는 웹 페이지를 제공해주는 웹 서버, 파일을 교환하는 FTP 서버, 이메일을 주고받을 수 있는 메일 서버(MX) 등이 있다. 웹 서버의 경우 클라이언트에게 요청한 해당 페이지의 HTML문서를 응답하여 해당 페이지를 클라이언트에게 보여주게 된다. ⁶

⁴ [인터넷, 컴퓨터인터넷IT용어대사전](#)

⁵ <https://www.liquidweb.com/blog/client-server-architecture/>

⁶ [월드와이드웹, 멀티미디어](#)

예를 들어 **대건고 미술 교사(이하 JP)**가 본인의 **네이버 카페**에 접속한다고 해보자. JP의 컴퓨터가 클라이언트, 네이버가 서버이며 서로간의 통신이 이루어 질것이다. JP가 **'대건고 미술교실'** 카페에 접속하기 위해 '네이버' 서버에 요청을 보낼 것이다. 서버는 요청에 문제가 없는지 확인한 후 클라이언트에게 대건고 미술교실 페이지에 대한 HTML 문서를 응답해주고, JP는 해당 HTML 문서를 웹 브라우저로 렌더링하여 눈으로 해당 페이지를 볼 수 있게 된다. 이것이 우리가 웹 페이지에 접속하여 페이지를 보고 인터렉션하는 과정이다.

위와 같은 과정을 위하여 필요한 요소가 몇가지 존재한다.

- **URL(Uniform Resource Locator):** 인터넷에서 특정 리소스의 위치를 가리키는 주소이다 ⁷
- **웹 브라우저(Web Browser):** 웹 페이지를 불러오고 렌더링 해주며 다양한 추가 기능을 제공해주는 인터넷과 컴퓨터를 연결해주는 소프트웨어. 대표적으로 구글 크롬, 네이버 웨일, MS 엣지 등이 있다 ⁸
- **하이퍼 텍스트(Hyper Text):** 어떠한 텍스트가 특정한 다른 요소를 참조하는 텍스트. 예를 들어 ⁹를 누르면 6번 각주로, '1. 서론'을 누르면 서론으로 이동하게 되며, <https://cafe.naver.com/artfeelclass> 누르면 '대건고 미술교실' 카페로 이동하게 된다. 특히 위와 같이 링크 형식인 경우 하이퍼 링크라고 부르며 웹은 이러한 하이퍼 링크들을 통해 서로 서로 이동하며 **서핑**을 할 수 있게 해준다. 하이퍼 링크를 다른말로 역링크, 백링크라고 부르기도 한다. 백링크는 논문에서의 피인용 수와 마찬가지로 그 페이지에 대한 **권위**를 나타낸다 ^{10 11}

⁷ [URL, MDN](#)

⁸ [웹 브라우저, 위키피디아](#)

⁹ 6번 각주를 의미하는 숫자 '6'을 클릭하면 6번 각주 내용으로 이동한다. 즉 6이란 텍스트가 6번 각주라는 리소스를 참조한다

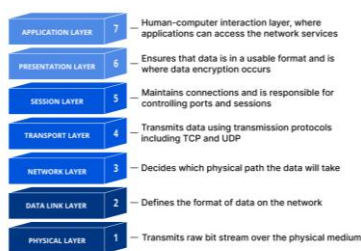
¹⁰ [웹, 소프트웨어 어휘다지기 - 중등](#)

¹¹ [백링크, 위키피디아](#)

2. URL(uniform resource locator)

클라이언트가 웹 페이지를 보기 위해서는 먼저 보고자하는 웹 페이지를 소유한 서버가 어디인지를 알아야 한다. 그런 다음 어떤 페이지를 보고싶은지를 알려주어야 한다. 이러한 내용을 담은 문자열을 URL(uniform resource locator)라고 부른다. 일반적으로 '프로토콜://서버네임/디렉터리네임/파일네임' 으로 구성된다.

0) OSI 모델



국제표준화기구(ISO)에서 개발한 모델로, 컴퓨터 네트워크와 통신 등을 계층을 나누어 개념을 분류해 놓은 모델이다. 이 중 3~5번 레이어가 각각 네트워크, 통신, 세션 레이어이며, 본 보고서에서 지속적으로 언급될 예정이다.¹²

1) 통신 프로토콜

컴퓨터와 컴퓨터 사이에서 데이터를 원활히 주고받기 위하여 약속한 여러 가지 규약¹³. 기기간 서로 통신을 하려면 서로 체계적인 규칙에 따라야 한다. 예를들어 A는 B에게 전화만 계속하고, B는 A에게 메시지만 계속 보낸다면 둘은 소통을 원활하게 하지 못할 것이다. 이러한 문제는 컴퓨터간 네트워크 통신에도 똑같이 적용되며, 이를 해결하기 위해 통신에 앞서 기기간 통신을 어떻게 수행할지 '규칙'을 정하고 이를 준수하며 통신한다. 어떤 두 컴퓨터 A와 B가 통신을 한다면 우선 어떤 방식으로 통신을 할 지 미리 약속된 규칙들 중 하나를 택하여 합의를 본 다음 해당 프로토콜에 따라 체계적이고 '규칙'적으로 통신하게 된다. 이때 이 약속된 '규칙'을 프로토콜이라 한다. 프로토콜은 공개적으로 모두가 사용할 수 있는 표준화된 공개 범용 프로토콜과 조직 내에서만 사용 가능하고 비공개되어 안전한 폐쇄 프로토콜이 있다.

¹² <https://www.cloudflare.com/ko-kr/learning/ddos/glossary/open-systems-interconnection-model-osi/>

¹³ [표준국어대사전](#)

공개 통신 프로토콜(OSI L3)에는 대표적으로 다음과 같은 프로토콜이 존재한다.¹⁴

- HTTP/HTTPS: 추후 **4. HTTP 프로토콜**에서 설명
- FTP: 파일 전송을 위한 프로토콜이다. 명령 채널에서 어떤 파일에 액세스 할 것인지를 지정하고 기본 정보를 전송한 다음, 데이터 채널에서 두 기기간 파일 데이터를 전송한다.¹⁵
- SSH: 원격 호스트에 접속하기 위한 보안 프로토콜이다. 기존 TelNet 프로토콜의 암호화 미지원으로 인한 보안 취약 문제를 해결하기 위해 개발되었다. 예를 들어 서버를 원격으로 호스팅할 때 클라이언트와 호스트(서버)간 인증 수단인 Key가 필요한데, 이 키를 생성할 때 SSH 프로토콜이 사용된다. 서버를 호스팅하면 호스트와 클라이언트가 대칭된 키를 발급받고(주로 .pem 파일) 이 파일을 이용하여 인증하면 서버에 접속하는 방식이다. 즉 쉽게 말하자면, 파일이 물리적인 '열쇠'이고 SSH가 잠금 장치라 하면, 문(호스팅 서버)을 열기위해 열쇠구멍(SHH)에 열쇠(Key)를 넣으면 문이 열리는 방식이다.¹⁶
- SSL: 추후 **3) SSL/TLS**에서 설명

¹⁴ [통신 프로토콜, 위키피디아](#)

¹⁵ <https://www.dropbox.com/ko/resources/what-is-ftp>

¹⁶ <https://library.gabia.com/contents/infrahosting/9002/>

2) IP

인터넷 프로토콜의 약자로 OSI L3 에 해당되는 네트워크와 관련된 프로토콜이다. 인터넷 유저는 수십억명으로 이들 각각을 구분하기 위해서는 고유한 식별 인자가 필요하다. 이를 위해 IP 주소를 모든 인터넷 유저에게 개별적으로 할당하여 유저들을 구분하고 관리한다.

IP 주소는 컴퓨터(네트워크)의 집 주소이다. 예를들어 구글의 경우 142.250.206.14 라는 IP 주소를 지니고 있다. 만약 google.com 에 접속하고 싶다면 142.250.206.14 라는 IP 주소에 방문하면 된다.

IP 주소의 전체 수는 한정되어 있어 주소의 부족 문제를 해결하고자 정적 IP 와 동적 IP 를 구분하여 사용된다. 일반적인 유저의 경우 자신의 컴퓨터로 인터넷에 접속하면 인터넷 제공자(ISP) 가 자신에게 할당 된 IP Pool 내에서 하나의 IP 주소를 유저에게 할당하게 된다. 또한 셀룰러 데이터를 사용하는 경우 기지국 별로 IP 주소가 할당되어 같은 기지국을 사용하는 유저들의 IP 주소가 동일할 수도 있다. 이러한 IP 주소를 동적 IP 라고 한다. 반면 Cloudflare 의 1.1.1.1 이나 Google 의 8.8.8.8 같이 대기업이나 특정 ISP 가 대가를 지불한 경우 고정된 IP 주소인 정적 IP 를 보유할 수 있다.

IP 주소를 표현하는 방식에는 IPv4 와 IPv6 가 있다. 각각 32 비트와 128 비트의 주소이다. IPv4 의 경우 $2^{32} = 4,294,967,296$ (약 43 억)개의 서로 다른 IP 주소가 존재하지만 인터넷 유저의 증가로 모든 IP 를 다 할당하여 부족해지자 IPv6 를 개발하여 사용하고 있다. IPv6 의 경우 $2^{128} = 340,282,366,920,938,463,374,607,431,768,211,456$ (약 340 **간...**)의 서로 다른 주소가 존재하여 고갈될 일이 절대 없다. IPv4 는 . 으로, IPv6 는 : 으로 비트를 구분하며, IPv4 는 123.456.789.100 과 같은 형식으로, IPv6 는 1234:5678:0910:aabb:ccdd:eeff 과 같은 형식으로 이루어져 있다. 한국의 경우 보유중인 IPv4 주소가 아직 남아서 IPv4 를 보편적으로 사용하고 있다. ^{17 18}

¹⁷ <https://namu.wiki/w/IP%20%EC%A3%BC%EC%86%8C>

¹⁸ <https://www.cloudflare.com/ko-kr/learning/dns/glossary/what-is-my-ip-address/>

3) 도메인 네임

도메인 네임(Domain name, 이하 도메인)은 웹에서 기억하기 힘든 IP 주소 대신 사용하는 텍스트 문자열이다. 앞서 설명하였듯 기존 IP 주소의 경우 복잡한 형태로 이루어져 있다. 만약 구글에 접속하고 싶는데 매번 URL창에 142.250.206.14를 기억하고 타이핑하고 있기에는 너무 비효율적이다. 이를 해결하기 위해 이 142.250.206.14 라는 IP주소를 도메인이라는 텍스트로 매핑한다. 유저는 구글에 접속할 때 142.250.206.14 대신 google.com을 타이핑함으로써 간결하고 외우기 쉬운 텍스트 문자열로 구글에 접속할 수 있게 된다.

도메인은 기본적으로 abc.def.ghi 형식으로 구성되어 있다. ghi에 해당되는 가장 후반부에 있는 도메인이 TLD(최상위 도메인)이다. 해당 도메인은 ICANN¹⁹와 미국, 그 외 각국 정부가 관리한다. TLD의 예시로는 com, net, kr, us, gov, org 등이 있다. def는 도메인의 가운데에 있는 2차 도메인²⁰이다. 2LD는 필수가 아니지만 조금 더 용도를 특정하기 위해 사용된다. 대한민국이 관리하는 kr도메인의 2차도메인 예시로는 co, hs 등이 있다. abc는 3차도메인이다. 주로 해당 도메인이 어떤 도메인인지 유저들이 쉽게 인지할 수 있도록 브랜드명이나 자신의 웹사이트를 잘 설명하는 동시에 외우기 쉽고 간결한 텍스트를 채택한다. 예를들어 google의 google이 있다. ^{21 22}

도메인을 도로명 주소²³로 비유해보자. 대건고등학교 주소인 '대구광역시 달서구 월곡로94길 46'에서 대구광역시는 TLD, 달서구는 2LD, 월곡로는 3LD라고 할 수 있다. 그 뒤 상세주소는 상세 경로(또는 4LD)로 생각할 수 있다.

¹⁹ 국제인터넷주소관리기구. 전세계의 도메인을 관리한다

²⁰ 기본적으로 도메인은 1차도메인(TLD)와 2차도메인(2LD)으로 끝이 난다(ex. google.com). 다만 여기서는 전체 도메인이 3차까지 있다 가정하고 설명하겠다.

²¹ <https://www.cloudflare.com/ko-kr/learning/dns/glossary/what-is-a-domain-name-registrar/>

²² [도메인 네임, 위키피디아](#)

²³ 앞서 IP주소를 집주소로 비유하였는데, IP주소에서의 '집주소'와 도메인에서의 '집주소'는 의미하는 바가 다르다. 굳이 따지자면 IP주소의 집주소는 우편번호를 의미한다고 할 수 있다.

도메인에서 상위 도메인 아래의 하위 도메인을 생성할 수 있다. 예를 들어 com의 소유주 Verisign사는 레지스트란트의 요청에 따라 2차 도메인을 생성하여 해당 도메인을 고객에게 제공한다. 만약 어떤 고객이 daegun.com 도메인을 구매했다고 해보자. 해당 2차 도메인은 해당 고객에게 소유권이 있으므로 해당 도메인 아래의 3차 도메인은 해당 고객이 새롭게 생성이 가능하다. 예를들어 해당 고객은 추가비용 없이 hs.daegun.com 이란 새로운 도메인을 생성하여 사용할 수 있다. 이때 daegun.com 과 hs.daegun.com과 ms.daegun.com 모두 **서로 다른 도메인**으로 취급한다. 물론 특정 3차도메인만 소유권을 양도할 수도 있다.

도메인은 ICANN - 레지스트리 - 레지스트라 - 레지스트란트로 구성되어 있다. ICANN이 대한민국이나 Verisign같은 레지스트리에 TLD를 할당하면 레지스트라를 통해 도메인을 판매한다. 레지스트란트는 레지스트라로부터 도메인을 임대하여 사용한다. 즉 ICANN으로부터 TLD를 레지스트리(실질적 관리인)에게 할당하고, 레지스트란트(실질적 사용자)는 레지스트라(중개업체)를 통해 도메인을 소정의 수수료를 지급하여 사용권을 부여받는다. ²⁴

IP주소를 도메인네임으로 매핑하려면 이를 중개하는 DNS가 필요하다. DNS에 대해서는 추후 **3. DNS(Domain Name System)**에서 설명하겠다.

²⁴ <https://namu.wiki/w/%EB%8F%84%EB%A9%94%EC%9D%B8%20%EB%84%A4%EC%9E%84#s-2>

4) 쿼리 스트링

쿼리 스트링(Query String, 또는 쿼리 파라미터)은 url 뒤에 추가적인 정보를 제공하는 방법이다. 주로 렌더링 될 페이지의 js 변수를 선언하고 값을 대입시킬 때 사용한다. 형식은 `daegun.hs.kr/?foo=abc&bar=123` 이다. 해당 쿼리 스트링의 의미는 abc라는 값을 지닌 foo 변수와 123이란 값을 지닌 bar 변수를 정의하라는 뜻이다.

네이버 검색을 예로 들어보자. 네이버에 대건고등학교를 검색하면, <https://search.naver.com/search.naver?query=대건고등학교>²⁵ 라고 나올 것이다. 이는 대건고등학교라는 값을 지닌 query변수가 새로 생성되었음을 의미하며 이 값을 기존 어떠한 변수에 대입하고 검색하는 로직을 거침으로써 대건고등학교에 대한 검색 결과가 출력되는 것이다.

쿼리 스트링은 동일한 페이지에 대해 어떤 값에 따라 출력 결과가 달라져야 하거나 어떤 특정 정보를 함께 제공해줘야 하는 경우 자주 사용된다. 예를 들어 어떤 커뮤니티 사이트의 게시판 url이 `board.daegun.com` 이라 치자. 12345번째 게시글을 표시하기 위해 `board.daegun.com/12345`와 같은 url을 사용할 수도 있지만, `board.daegun.com/?id=12345` 같은 url을 사용할 수도 있다. 둘의 차이점은 전자의 경우

`board.daegun.com/12345`와 `board.daegun.com/67890`의 경우 서로 다른 페이지로 인식하지만, 후자의 경우 `board.daegun.com/?id=12345`와 `board.daegun.com/?id=67890`를 같은 페이지로 보게된다. 실제 서버에도 전자의 경우 각 게시글에 따른 폴더(또는 파일)가 개별적으로 존재하지만, 후자의 경우 하나의 폴더(또는 파일)를 통해 다양한 게시글을 보여줄 수 있다.²⁶

쿼리 스트링은 세부 디렉터리를 포함한 모든 url을 작성한 뒤 `/?` 이후에 작성한다.

²⁵ 실제로는 뒤에 이전 쿼리나 기타 다양한 정보들이 추가로 생성되지만, 이해를 위해 간략화 하였다

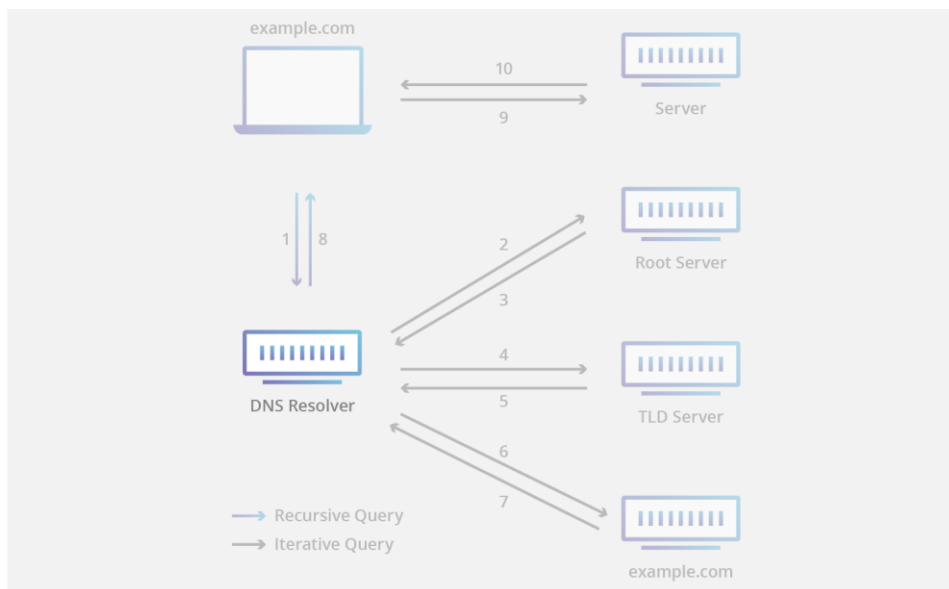
²⁶ 비유하자면 전자는 종이책, 후자는 태블릿의 E-Book이라 할 수 있겠다

3. DNS(Domain Name System)

DNS란 앞서 언급한 기존 IP주소로 이루어진 URL을 유저가 도메인을 통해 접속할 수 있도록하는 시스템이다. 기존 IP주소의 문제점들을 해결하기 위해 이 IP주소에 따른 도메인네임을 설정하고, 만약 누군가가 해당 도메인네임을 URL로 입력하게 되면, DNS 서버는 도메인에 해당하는 IP주소를 반환함으로써 유저가 원하는 사이트에 액세스 할 수 있도록 한다.

기본적으로 중개 역할 및 캐싱(트래픽이 높은 사이트들의 DNS를 자체적으로 저장) 하는 ISP(통신사, SKT/KT 등)가 각각 소유하는 **DNS 리커서**, 모든 TLD에 대한 네임서버 주소를 보유중인 ICANN 소유의 **DNS 루트 서버**, 각 TLD에 따른 각 사이트의 IP주소를 알고 있는 권한 있는 네임 서버들의 IP 주소를 보유중인 Verysign 또는 국가정부 등이 각각 소유하고 있는 **TLD 네임서버**, 해당 도메인의 실질적인 IP주소를 보유중인 **권한 있는 네임 서버**로 구성되어 있다. ²⁷

다음은 어떤 사용자 A가 google.com 에 접속하기 위해 DNS로부터 IP주소를 얻어내는 과정을 간략히 나열한 것이다.



28

²⁷ <https://aws.amazon.com/ko/route53/what-is-dns/>

²⁸ <https://www.cloudflare.com/ko-kr/learning/dns/dns-server-types/>

1. google.com에 접속하고 싶다는 신호가 ISP로 인터넷을 통해 전달된다
2. ISP의 DNS 리커서²⁹는 해당 도메인에 대한 IP주소를 얻기 위하여 ICANN이 운영하는 DNS 루트 서버에 전달한다
3. A는 google.com에 접속하고 싶으므로 DNS 루트 서버가 해당 TLD(.com) 네임 서버에 전달한다
4. .com의 TLD 서버는 해당 .com 도메인의 IP주소를 가지고 있는 권한있는 네임서버에 전달한다
5. 권한 있는 네임 서버(구글 또는 호스팅 업체가 운영)는 해당 도메인의 실질적인 IP주소를 반환한다 ³⁰

위 설명에서 볼 수 있듯 최종적으로 유저 A가 원하는 google.com의 IP주소는 '권한 있는 네임 서버'만이 가지고 있다. ^{31 32}

그러나 해당 IP주소를 얻기 위해 총 4개의 DNS 서버를 거친다. 이러한 매커니즘은 비효율적으로 보인다. 원활한 이해를 위해 이번엔 실무적인 관점에서 보자. 웹 개발자 B가 사이트를 개발하여 도메인을 구매하였다고 해보자. 기존 IP주소 123.456.789.000을 helloworld.com 으로 매핑한다고 했을 때, B는 이를 위해 먼저 자신이 도메인을 구매한 레지스트라 또는 서버 호스팅 업체 또는 기타 다른 업체의 네임 서버에 자신의 도메인과 IP주소를 연결하여 등록한다. 이때 이 네임 서버가 **권한 있는 네임 서버**가 된다. 위에서 설명한 계층적 구조 덕분에 권한 NS를 제외한 타 서버들은 새로운 도메인과 IP주소에 대해서 알고 있을 필요도 이유도 없다. 그저 권한 NS의 주소만 알고 있다면 IP주소는 권한 NS가 반환해주기 때문이다.

²⁹ 재귀확인자. 1차적으로 유저의 요청을 받는 중개인 역할이다. 일반적으로는 요청을 받으면 각각의 서버들과 통신하여 IP주소를 찾아내지만, 만약 최근 누군가 해당 사이트에 접속하여 캐싱된 기록이 있을 경우 타 DNS 서버를 거치지 않고 바로 IP 주소를 반환할수도 있다

³⁰ 모든 DNS 서버들은 중개자인 DNS 리커서와만 통신하지만 생략했다

³¹ <https://www.cloudflare.com/ko-kr/learning/dns/what-is-dns/>

³² <https://www.cloudflare.com/ko-kr/learning/dns/dns-server-types/>

이렇듯 DNS 가 저런 복잡한 매커니즘을 지닌 이유는 효율성 때문이다. 인터넷의 매시간 급변한다. 수시로 서버의 IP 주소가 변동되는가 하면 새로운 사이트가 생기거나 어느날 갑자기 사라지기도 한다. 그럴 때 마다 모든 DNS 서버가 일일이 해당 변화에 맞춰서 모든 도메인을 업데이트 하기엔 너무 비효율적이다. 이러한 이유로 계층적으로 분리하여 역할을 분담하여 최종적으로 네임 서버들만 자신드링 관리하는 도메인의 변화만 관리하도록 하여 효율성을 추구하는 방식을 채택한것이다. 즉 리커서는 루트서버의 IP 주소만, 루트서버는 각 TLD 들의 IP 주소들만, 각 TLD 의 네임서버는 자신들의 TLD 를 사용하는 하위 네임서버들의 IP 주소들만, 해당 권한있는 하위 네임서버는 자신들의 NS 에 등록된 도메인의 IP 주소만 알고 있으면 유저는 세상의 모든 사이트³³의 IP 주소를 도메인만 알고 있다면 접속할 수 있다.

또한 DNS 는 단순히 도메인을 IP 주소로 변환해주는 역할만 수행하는 것이 아니다. 각각의 DNS 서버들은 하위 DNS 서버의 트래픽을 고려해 서로 같은 역할을 하는 서버들은 번갈아 가며 반환하여 트래픽 편향을 줄여주며, DNS 리커서를 포함하여 DNS 서버들은 트래픽이 상대적으로 많은 도메인의 IP 주소를 미리 캐싱하여 하위 DNS 서버를 거치지 않고 바로 IP 주소를 직접 반환해주기도 하며, DNSSEC(DNS Security Extensions)를 통해 위변조 응답이나 DNS 하이재킹 등의 해킹을 방지하며, 악성 웹사이트의 실제 IP 주소 대신 안전한 주소(한국의 경우 대표적으로 <http://warning.or.kr/>)로 임의 변경(하이재킹과 겹으로 비슷한 원리이다)하여 사용자를 보호(또는 검열..) 한다. ³⁴

대표적인 DNS 리커서(재귀 확인자)는 각 통신사별 리커서 및 구글의 8.8.8.8 과 ColudFlare 의 1.1.1.1 이 있다. 신뢰가능한 리커서를 사용해야 보안이나 해킹으로부터 안전하다.

³³ 자사 인트라넷이나 프라이빗 도메인, Tor의 onion 등 특수한 보안 사이트는 제외. 해당 사이트들은 각자 지정된 방식과 프로토콜로 독자적인 DNS를 지니고 있다

³⁴ <https://aitown.tistory.com/1076>

4. HTTP 프로토콜

앞서 '1) 통신 프로토콜'에서 설명한 프로토콜 중 대부분의 사이트 라우팅 및 웹 통신에 사용되는 HTTP와 HTTPS 및 SSL에 대해 자세히 알아보겠다.

1) HTTP(Hypertext Transfer Protocol)

HTTP(Hypertext Transfer Protocol)는 영국의 팀 버너스-리³⁵가 설계한 클라이언트-서버 간 통신을 위한 프로토콜이다. HTTP는 미리 약속된 Method³⁶를 통하여 HTML 문서를 전송하기 위한 프로토콜로, 클라이언트가 서버에 특정 정보(HTML 등 웹과 관련된 데이터)를 요청하면 서버는 해당 정보를 클라이언트에게 응답해주게 된다.³⁷ 이때 클라이언트는 HTTP 통신을 위해 크롬 등의 웹 브라우저를, 서버는 요청을 해석하고 응답하기 위해 Apache, nginx 등의 서버 소프트웨어를 사용한다.

HTTP는 Connectless 방식을 채택하여 사용한다. Connectless란 말 그대로 서버가 클라이언트에게 응답을 하면 연결을 끊는다. 서버는 연결을 끊어버렸으므로 응답을 받은 클라이언트의 이후 상황(State)를 알 수 없게된다(Stateless). 만약 추가적으로 서버와의 통신이 필요한 경우 다시 HTTP로 연결하여 통신하고 해제하고를 반복하게 된다. 이러한 방식은 불특정 다수가 지속적으로 접속하여 다량의 트래픽이 유발되는 웹 서버에게 적절한 방식이다. 연결을 끊어 트래픽 유지 시간을 최소화하므로 좀 더 적은 리소스로 많은 클라이언트에게 페이지를 라우팅 할 수 있게 된다.

³⁵ **Timothy John Berners-Lee**. 1955년생 영국의 컴퓨터 엔지니어. WWW, URL, HTTP등을 고안하여 '웹의 아버지'라고 불림([팀 버너스리, 나무위키](#))

³⁶ HTTP Method 및 HTTP 요청을 보내는 자세한 방법에 대한 내용은 [https://github.com/jjy0809/creanpl/blob/main/인터넷 닷글 검열 확장프로그램 개발.pdf](https://github.com/jjy0809/creanpl/blob/main/인터넷%20닷글%20검열%20확장프로그램%20개발.pdf) 를 참고하기 바란다

³⁷ <https://ko.wikipedia.org/wiki/HTTP#>

그러나 만약 웹 사이트가 사용자와 인터랙션하는 경우 문제가 발생한다. 연결을 끊어버려 클라이언트의 상태를 알 수 없으므로 로그인이나 진행 상황 등 '기억' 할 필요가 있는 내용들도 모두 잊어버리게 된다. 이를 해결하기 위해 쿠키를 사용한다. ³⁸

2) HTTPS(Hypertext Transfer Protocol Secure)

HTTPS는 기존 HTTP의 보안 강화 버전으로, 통신의 인증과 암호화를 위해 넷스케이프(Netscape)³⁹가 개발하였다. 기존 HTTP의 경우 클라이언트와 서버 간 연결에서 암호화가 적용되지 않았지만, HTTPS의 경우 SSL/TSL를 통해 오고가는 데이터들을 암호화하여 안전하게 통신할 수 있게 해준다. 덕분에 HTTPS는 기존의 일반적인 PlainText로 데이터를 주고받아 아무나 네트워크에 침투하여 데이터들을 가로챌 수 있는 HTTP와 달리 SSL을 통해 모든 데이터들을 암호화 하여 제3자가 침투해도 해당 데이터를 디코딩할 수 없어 보안이 강력하다. 또한 이러한 보안성 덕분에 검색 엔진은 HTTP보다 HTTPS를 적용한 사이트를 더 신뢰하여 사이트의 순위를 HTTPS를 더 높게 책정한다. 이는 트래픽 및 수익과 직결되는 중요한 장점이다. ⁴⁰

³⁸ https://www.joinc.co.kr/w/Site/Network_Programing/AdvancedComm/HTTP

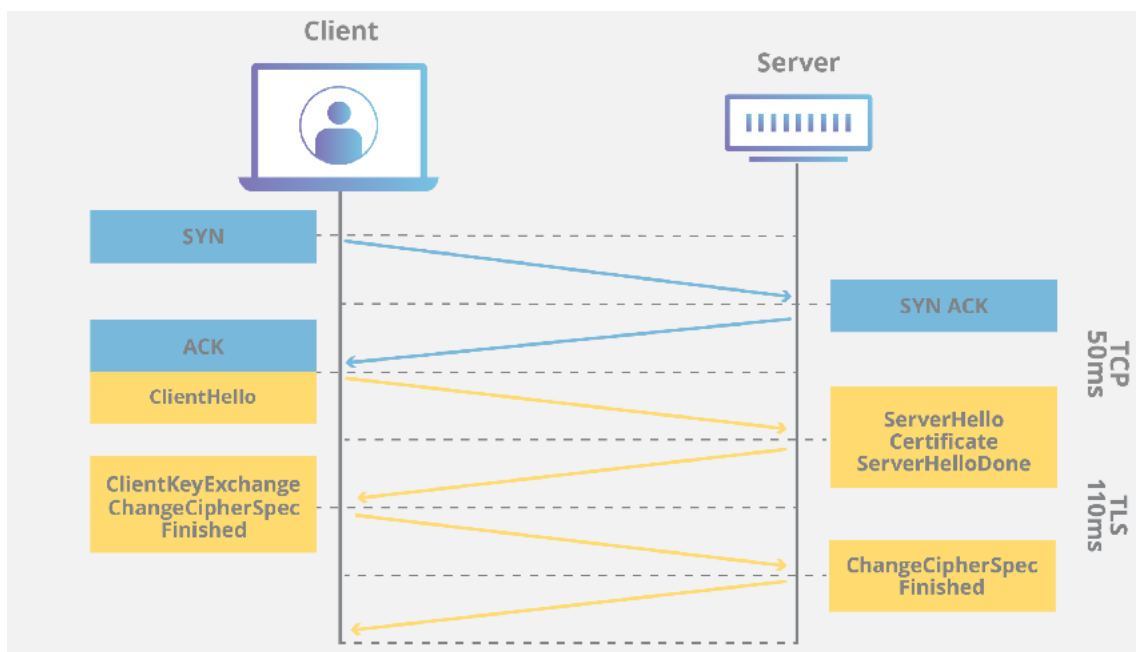
³⁹ 1900년대 점유율 90%의 웹브라우저. 그러나 1990년대 후반, 마이크로소프트의 IE(인터넷 익스플로러)와의 웹 브라우저 전쟁에서 패배하고 지원 중단됨. 이후 개발자들이 모질라로 분리하여 파이어폭스 브라우저를 개발하여 현재도 상용되고 있음 ([넷스케이프](#), [위키피디아](#))

⁴⁰ <https://aws.amazon.com/ko/compare/the-difference-between-https-and-http/>

3) SSL/TLS

SSL(**Secure Sockets Layer**) 또는 TLS(**Transport Layer Security**)⁴¹은 인터넷 상에서 통신을 하기 위해 데이터를 암호화하는 보안 프로토콜이다. SSL은 데이터를 암호화하고(암호화), 수신자가 요청자임을 보장하고(인증), 데이터가 위/변조 되지 않도록 확인하는(무결성) 역할을 수행한다. 위에서 설명한 HTTPS는 이 SSL을 HTTP에 적용시켜 보안을 강화한 사례이다. SSL을 적용하기 위해서 해당 웹사이트 서버에 SSL 인증서가 존재해야 하며, 이 인증서를 통해 인증 과정을 거친 다음 이후 연결을 HTTPS로 유지한다. ⁴² ⁴³

다음은 SSL을 이용하여 클라이언트가 HTTPS 사이트에 접속하는 과정⁴⁴이다.



⁴¹ SSL에서 TLS로 명칭이 변화하였다. 즉 둘은 같다. 보편적으로 아직 SSL이란 용어를 더 많이 사용한다

⁴² <https://aws.amazon.com/what-is/ssl-certificate/>

⁴³ <https://www.cloudflare.com/ko-kr/learning/ssl/transport-layer-security-tls/>

⁴⁴ 이 과정을 핸드셰이크(handshake)라고 한다

1. 클라이언트가 접속하고자 하는 웹 사이트의 서버에 클라이언트가 지원하는 SSL 버전, 지원되는 암호 제품군, 서버 도메인, 세션 식별자, 암호(무작위 문자열) 등이 포함된 'ClientHello' 메시지를 전송한다
2. 서버는 서버의 SSL 인증서, 서버가 채택한 SSL 버전, 암호 등이 포함된 'ServerHello' 메시지로 응답한다
3. 클라이언트는 응답받은 SSL 인증서를 해당 인증서를 발행한 CA(발행기관)에 감정을 요청하여 해당 서버가 안전한지 확인한다
4. 안전이 확인된 후 클라이언트는 SSL 인증서로부터 Public Key를 받고, 이 Key를 이용하여 예비 마스터 암호를 무작위로 생성하여 암호화⁴⁵한 다음 서버에 전송한다
5. 통신 상대가 당사자임을 확인하기 위해 클라이언트와 서버 모두 총 3개의 암호를 이용하여 세션키를 생성한다. 이때 둘의 세션키가 동일하다
6. 서로의 세션키로 'ChangeCipherSpec' 메시지를 보내 서로의 세션키가 동일함을 확인하고, 핸드셰이킹 과정이 정상적으로 끝이 났음을 알린 다음, 본격적으로 세션키만을 사용하여 HTTPS 연결을 유지하고, 데이터를 암호화하여 통신하게 된다 ^{46 47}

핸드셰이킹을 쉽게 설명하자면 서로간만 아는 비밀번호를 만들어내는 과정이다. 이 과정에서 SSL인증서와 Public Key 및 Private Key가 필요하며, 이를 통해 비대칭 암호화로 안전하게 서로만의 비밀번호인 Session Key를 생성한다.

⁴⁵ HTTPS 연결 과정에는 대칭 암호화와 비대칭 암호화 두가지 과정이 존재한다. 이경우는 비대칭 암호화이며 핸드셰이크 과정이 끝난후 실질적인 암호화는 대칭암호화를 사용한다. 핸드셰이크에서 클라이언트는 Public Key만, 서버는 Public Key와 Private Key를 소유하고 있다. Public으로 암호화된 데이터는 Private로만 복호화 할 수 있으며, Private로 암호화 된 데이터는 Public으로만 복호화 할 수 있다. 클라이언트는 암호화 및 복호화에 Public Key를, 서버는 Private Key를 사용하게 된다. ([AWS](#)) 그러나 이후 세션키를 클라이언트와 서버 모두 소유하게 되었을 경우 둘은 서로 대칭되는(동일한) 세션키만을 이용하여 암/복호화 한다. 즉 세션키를 만드는 과정은 비대칭 암호화를, 세션키를 생성한 후에는 대칭 암호화를 사용한다

⁴⁶ <https://www.cloudflare.com/ko-kr/learning/ssl/what-happens-in-a-tls-handshake/>

⁴⁷ <https://namu.wiki/w/TLS#s-2>

이러한 복잡한 과정을 보면 굳이 세션키를 생성할 필요가 있나 싶을 수 있다. 이미 Public key와 Private Key 만으로도 얼마든지 안전하게 데이터를 암호화 할 수 있기 때문이다. 그러나 이런 핸드셰이킹 과정을 거치는 이유는 비대칭 암호화보다 속도와 리소스 관리 측면에서 훨씬 우수하기 때문이다. 서로 대칭되는 키를 가지고 암복호화를 하므로 당연히 연산량이 줄어들어 리소스를 덜 잡아먹고 속도도 훨씬 빠른것이다. 또한 이미 인터넷에 공유된 Public Key보다 죽석으로 랜덤하게 만든 Session Key가 더욱 안전하다. 이러한 점 때문에 이런 복잡한 핸드셰이킹 과정을 거치는 것이다.

이러한 SSL 인증서를 발급해주는 인증된 발급 기관(CA)이 존재하며, 각자의 서버와 브라우저 모두 신뢰 가능한 CA 목록을 지니고 있다. 이 신뢰 가능한 CA에서 발급한 SSL 인증서를 서버가 보유해야 HTTPS 연결이 가능해진다. 사이트 소유주는 이 CA를 통해 SSL 인증서를 최대 13개월의 유효기간으로 발급받을 수 있으며, 가격⁴⁸에 따라 HTTPS 통신 도중 해킹이 발생 할 경우 CA에서 보상금을 지불하기도 한다.

HTTPS의 기본적인 작동 원리를 이해하는 것이 목적이므로 디지털 서명이나 신뢰 사슬 등 더 자세한 내용은 생략하겠다

⁴⁸ 무료 SSL 인증서도 존재하며 대부분 3개월에 만원부터 시작해서 최대 1000만원까지 올라간다

5. 적용

위에서 설명한 이론을 바탕으로 실제로 웹 사이트에 어떻게 적용되는지를 컴퓨터가 웹 사이트에 접속하는 과정 예시를 통해 알아보겠다. DNS의 원리 및 HTTPS 프로토콜을 이용한 암호화 통신은 앞서 설명했으므로 생략한다.

대건고등학교 학생 A가 미술 수업에서 자신의 과제물을 제출하기 위해 [‘대건고 미술교실’](#)에 방문한다고 가정하자. 대건고 미술교실 사이트는 네이버 카페를 통해 만들어 졌으며, 도메인 주소는 <https://cafe.naver.com/artfeelclass>이다. 학생 A는 컴퓨터를 통해 크롬(또는 타 브라우저)를 실행한 다음, 주소창에 위 도메인 주소를 입력할 것이다. 학생 A의 크롬은 해당 도메인을 토대로 A에게 대건고 미술교실 카페를 접속할 수 있게 해줘야 한다. 먼저 크롬은 도메인을 확인한다. TLD가 com, 2LD가 naver, 3LD가 cafe인 것을 파악하고 이를 DNS 서버에 전송한다. DNS 서버는 해당 도메인에 해당하는 IP 주소를 반환한다. 크롬은 해당 IP주소에 해당하는 서버에 요청을 보낸다. 이때 이 요청은 url의 프로토콜, 즉 위 예시에서는 https 프로토콜을 이용하여 요청을 보낸다. 만약 A가 <http://cafe.naver.com/artfeelclass>를 url로 입력했다면 크롬은 http 프로토콜을 이용하여 요청을 보낼 것이다⁴⁹. 서버측은 해당 요청이 정상적인지를 판단한 후 어떤 페이지를 반환해야 하는지 판단할 것이다. 이때 이 판단은 세부 디렉터리를 통해 파악한다. 기본적으로는 네이버 카페 ‘홈’을 반환한다. 그러나 url에서 보이듯 [café.naver.com](#) 뒤에 [/artfeelclass](#)가 존재한다. 이는 네이버 카페 서버의 artfeelclass 폴더에 들어있는 페이지 코드에 접속하고 싶다는 뜻이다. 서버는 A에게 네이버카페 서버의 artfeelclass 카페에 대한 html, css, js 코드를 제공할 것이다. 또한 A는 해당 코드를 크롬을 통해 렌더링하여 페이지를 비로소 볼 수 있는 것이다.

⁴⁹ 대부분의 권위있는 사이트는 https를 사용하며 http 요청도 강제로 https를 적용시킨다. 실제로 위 http url을 클릭해도 최종적으로 https로 연결된 대건고 미술교실 카페에 접속하게 된다. 이는 해당 서버의 정책에 따른 것이며 아예 https를 지원하지 않거나 둘 다 지원하는 사이트도 존재한다

III. 웹 프로그래밍

이제 웹을 개발하는 방법을 알아보겠다. 중학생때 이미 웹 개발을 배우고 해본 적 있지만 시간이 많이 흘렀음으로 다시 공부 할 필요가 있다고 느꼈고, 이를 위하여 도서 '[코딩 자율학습 HTML + CSS + 자바스크립트\(김기수 저, 길벗\)](#)'⁵⁰를 읽어보고 직접 실습해보며 웹 개발을 익혔다. 분량상 HTML, CSS, JS가 무엇인지만 간단하게 언급하고 세부적인 내용은 결과물로 대체하겠다.

1. HTML

HTML(**H**yper **T**ext **M**arkup **L**anguage)이란 웹 브라우저의 페이지를 표시하도록 설계된 마크업 언어⁵¹이다. ⁵²

먼저 HTML은 기본적으로 다음과 같이 구성되어 있다.

`<태그 속성="값">콘텐츠</태그>`

먼저 태그는 웹 페이지를 구성하는 요소를 정의하는 역할로 HTML 문법을 이루는 가장 작은 단위이다. 각 용도에 맞게 div, span, p, a, img, iframe 등 매우 많은 태그들이 존재하며 각 브라우저와 버전에 따라 기본적으로 지원하는 태그도 다르며, css나 js 코딩을 통해 새로운 태그를 만들어서 사용하는것도 가능하다.

속성은 태그에 어떤 의미나 기능을 보충하는 역할이다. 쉽게말해 옵션이다. 태그에 이름(class 또는 ID)을 붙여주거나, 어떤 기능을 추가해주기 위해 사용한다. 기본적으로 모든 태그에 class나 id, style 태그 등을 추가할 수 있고, 용도에 따라 href, type, placeholder, onclick 등 다양한 옵션을 속성으로 추가할 수 있다.

⁵⁰ 본 보고서에서 특정 내용의 출처가 해당 도서인 경우 출처 표기를 생략한다

⁵¹ 태그를 통해 구조를 명기하는 언어. 참고로 HTML은 프로그래밍 언어가 아니다

⁵² <https://ko.wikipedia.org/wiki/HTML>

태그는 크게 콘텐츠가 있는 문법과 없는 문법으로 나뉜다. 먼저 콘텐츠가 존재하는 경우 open 태그와 close 태그로 해당 콘텐츠를 감싼다. 이때 이 태그와 콘텐츠를 모두 합친 하나의 코드를 Element(요소)라고 한다. 반면 콘텐츠가 존재하지 않는 경우 close 없이 open만 사용한다. 예를 들어 들여쓰기를 의미하는
이 존재한다.

HTML의 기본 구조는 DTD, head, body로 구성된다. 먼저 DTD(Document Type Definition)는 웹 브라우저가 해당 HTML 문서를 어떻게 처리해야 할 지 알려주는 문구이다. 일반적으로 HTML5 형식으로 문서를 해석하며 HTML문서 최상단에 다음과 같이 기입한다.

```
<!DOCTYPE html>
```

head 태그는 HTML 문서의 메타데이터를 정의하는 영역이다. 인코딩 형식, 파비콘, 페이지 제목과 설명 등을 입력할 수 있다. body 태그는 본격적인 페이지, 즉 브라우저에 노출되는 내용을 작성하는 공간이다.

2. CSS

CSS(Cascading Style Sheets)는 페이지의 디자인을 위한 스타일 언어이다. 기본적으로 다음과 같이 구성되어 있다.

선택자 {속성: 값;}

먼저 선택자는 HTML의 어떤 요소(태크)를 디자인 할 것인지 지정해주는 역할을 한다. 어떤 태그 전체를 선택하거나 특정 클래스 또는 ID만 선택하거나, 어떤 요소의 자식 요소 중 특정 태그만을 선택하거나 등 유연하게 HTML내에 있는 요소들을 자유자재로 선택할 수 있다. 만약 클래스를 선택할 땐 선택자 앞에 . 을 붙이고, ID를 선택할 땐 #을 붙인다. 또한 자식을 선택하려면 부모 선택자와 자식을 > 로 잇고, 하위 요소만을 선택하고 싶으면 띄어쓰기로 구분한다. 또 여러 속성을 동시에 선택하려면 , 를 사용하면 된다.

선택자에 추가적으로 가상 클래스를 추가할 수 있다. 선택자:가상클래스 형식으로 선택 가능하며, 해당 요소가 어떤 특정한 조건을 만족했을 때 만 해당

스타일이 적용되도록 해준다. 예를 들어 마우스를 요소 위에 올렸을때를 감지하는 `hover`이나 해당 요소가 비활성화(disabled)되었을 때 작동하는 `:disabled` 등의 가상클래스가 존재한다.

속성의 경우 다양한 속성이 존재하며, 특정 태그 전용으로 작동하는 속성도 존재한다. 속성은 태그의 기본 스타일(ex `h1`의 경우 기본 텍스트보다 크고 볼드체로 표시됨)을 무시하고 override한다. 또한 `css` 내에서도 같은 요소를 선택한 코드가 여러 개라면 그 중에서 더 세부적으로 요소를 지정한 코드(ex `#foo > .bar` 보다 `p > #foo > .bar` 가 더 높은 우선순위를 지님)를 우선으로한다. 만약 특정 속성만을 최우선순위로 적용하고 싶다면 속성뒤에 `!important`를 작성하면 해당 스타일을 무조건 강제 적용하게 된다. 특정 요소의 자식 요소들은 부모의 스타일 속성을 상속받아 적용된다. 자식 요소에 특별히 설정된 속성이 없다면 부모의 속성을 그대로 상속받게 된다. 속성을 설정할 때 단위를 지정해줘야 하는 경우가 있다. 글자의 크기를 `50px`이라 할 수도 있지만, 유저의 스크린 크기에 따라 유동적으로 폰트 사이즈를 변경하고 싶다면 `%`, `포`, `vw`, `em`, `rem` 등의 상대 단위를 사용할 수 있다.

3. JS

JS(JavaScript)는 웹 페이지를 동적 웹으로 만들어 주기 위한 언어이다. 기존의 HTML/CSS만 적용된 웹 페이지는 단순히 요소들을 보여주지만 하여 유저와의 인터랙션이 불가하다. 그러나 여기에 JS를 넣어서 코딩해준다면 유저와 인터랙션하여 더욱 퀄리티 높고 다양한 기능을 수행할 수 있는 웹 페이지를 만들 수 있다.

JS에서는 기본적으로 변수⁵³가 두가지 버전이 있다. 바로 변수와 상수이다. 변수는 한번 선언⁵⁴한 뒤에도 해당 변수의 값을 직접 수정할 수 있다. 반면 상수의 경우 한번 선언되면 그 이후 코드를 통해 변수값을 변경하려고 해도 애러가 발생하고 변경이 거부된다. 상수가 존재하는 이유는 보안 때문이다. 일반적인 프로그래밍 언어에는 상수가 따로 존재하지 않고 상수도 변수로 선언하지만, 웹 특성상 변수값을 유저가 console을 통해서 쉽게 수정할 수 있으므로 사이트에 대한 공격이나 edge case를 미연에 방지하고자 상수와 변수를 언어 내부적으로 구분한다. 변수를 선언할 때 변수는 `var foo = 0;` 또는 `let foo = 0;`⁵⁵ 상수는 `const BAR`⁵⁶ `= 0;` 형식으로 코드를 작성한다.

⁵³ 프로그래밍에서는 기본적으로 상수도 변수라 부른다. 변수의 특성중에 상수와 변수가 존재하며 둘 다 같은 변수이지만 여기서 사용한 문맥상 두 변수의 의미는 조금 다르게 사용되었다

⁵⁴ 엄밀하게는 선언/할당/초기화로 구분된다. 변수의 식별자(var, let, const)를 지정하는 행위는 선언, 선언된 변수에 값을 대입하는 것을 할당, 선언과 할당을 동시에 하면 초기화한다고 한다. 그러나 본 보고서에서는 모든 모든 행위를 선언으로 통일하여 사용하겠다 (사실 도서에서는 이렇게 구분지어 봤지만, 업계에서 관례적으로 선언 또는 정의라는 용어를 많이 사용한다. 일부 언어의 경우 식별자 지정이 불가하여 선언만 할 수 없는 경우도 있으며, 굳이 구분하지 않아도 다 이해하여 의사소통에 문제가 없다)

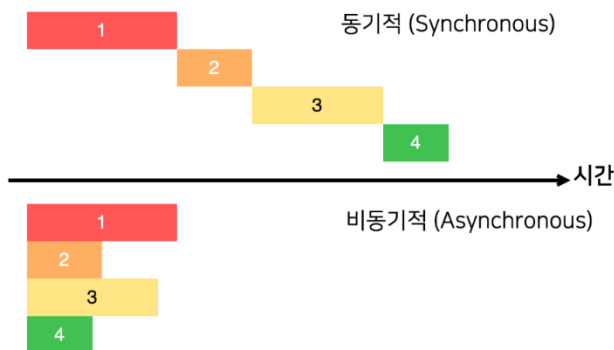
⁵⁵ let은 var의 업그레이드 버전으로 일부 기능들이 개선되었다. 중복 선언을 방지하고, 스코프(요소의 영역?)로 스코프 내부에 있는 요소끼리는 서로 공유되고 참조가 가능하다)가 살짝 다르다. 전문가들은 let 사용을 권장하지만, 업계에서는 여전히 혼용하여 사용되고 있다

⁵⁶ 관례적으로 상수는 대문자로 표기한다. 이는 상수를 지원하지 않는 언어에서 해당 변수가 변수인지 상수인지를 구분하기 위함이다

1) 비동기 함수

해당 내용은 도서에서는 언급만 하고 설명되진 않은 심화 내용이지만 웹 개발과 API 통신에서 중요한 개념이라 따로 설명하고 넘어가겠다

일반적인 프로그래밍 언어들은 코드를 위에서 아래로 순차적으로 실행한다. 즉 위의 코드가 모두 끝나면 다음 코드가 실행되는 방식이다. 그러나 웹 페이지를 로드하는데 코드를 하나하나 위에서부터 실행하고 있자니 속도가 매우 느리고 비효율적으로 느껴질 것이다. 이러한 문제를 해결하고자 비동기 함수가 존재한다. 기존의 동기함수는 코드가 순차적으로 실행되었다면, 비동기 함수는 코드를 동시에 실행한다.



비동기를 사용하는 이유는 다음과 같다.

1. 반응성 향상: 사용자가 요청한 작업이 완료될 때까지 기다리는 것은 비효율적이며 UX에도 좋지 못하다. 또한 네트워크 통신(API 통신)이 필요한 경우 필요한 데이터를 응답 받을 때까지 다른 작업을 수행할 수 없으므로 반응성이 떨어지게 된다
2. 속도 향상: 비동기를 통해 여러 작업을 동시에 병렬적으로 처리하게 된다면 시간이 오래걸리는 작업들을 동시에 작업하여 전체 작업 시간을 단축할 수 있다
3. 에러 처리: 비동기는 콜백과 Promise가 존재하여 에러 관리가 용이하다

위와 같은 장점 덕분에 웹코딩에서 JS 의 대부분의 함수는 비동기 함수로 코딩하게 된다. ⁵⁷

⁵⁷ <https://trustmitt.tistory.com/85>

그러나 비동기 함수에서 특정 코드가 순차적으로 동작해야 하는 경우엔 문제가 발생한다. 만약 이커머스 사이트에서 어떤 유저가 사과를 구매한다고 할 때, 결제 시스템을 비동기 함수로만 설계한다면 결제가 완료되지도 않았는데 사과가 주문이 되고, 결제는 잔액 부족으로 실패해버리는 일이 벌어질 수 있다. 이러한 문제들을 해결하기 위해 Promise 가 탄생하였다. 기존에는 비동기 함수에서의 순차적 작업을 위해 callback ⁵⁸ 함수가 사용되었다. 그러나 이러한 callback 함수는 가독성⁵⁹이 매우 나빠 보기에 좋지 못하다. 이러한 문제를 해결하기 위해 callback 과 동일한 역할을 하는 새로운 매킴니즘인 promise 를 만들게 되었다. promise 는 어떤 작업에 대한 상태를 나타내는 객체이다. promise 객체는 특정 작업의 상태를 나타내기 위해 다음 3 가지 상태를 지닐 수 있다.

- pending: 현재 작업이 진행중
- fulfilled: 작업이 성공적으로 완료됨. 이때 promise 오브젝트는 작업의 성공 결과도 함께 보유한다
- rejected: 작업이 어떤 이유로 실패함. 이때 작업의 실패 이유도 함께 보유함

이를 통해서 promise 오브젝트의 상태를 통해 완료될 때까지 기다릴지(pending), 다음 작업으로 넘어갈지(fulfilled), 넘어가지 말고 예외 프로세스를 실행할지(rejected)를 결정할 수 있다. promise를 쉽게 설명하기 위해 앞선 예시를 사용해보자. 만약 유저가 사과를 구매하려고 하는데 결제가 아직 진행중이면 promise오브젝트는 pending 상태를, 결제가 성공적으로 완료되었으면 promise 오브젝트는 fulfilled 상태와 결제 정보(카드 번호, 날짜, 금액, 기타 고유 번호 등등)를, 결제가 실패하였다면 rejected 상태와 실패 사유(잔액 부

⁵⁸ 이것이 무엇인지는 내용이 너무 복잡하고(콜백을 설명하려면 함수, 오브젝트, 파라미터 등의 매킴니즘부터 설명해야한다) 본 탐구에서 중요하지 않으므로 생략하겠다

⁵⁹ 프로그래밍에서 코드의 가독성은 매우 중요하다. 단순히 예뻐보이기 위함이 아닌, 개발자(특히 팀으로 협업하는 경우)가 해당 코드를 눈으로도 쉽게 이해할 수 있도록 함이다. 이는 장기적인 프로젝트나 유지/보수에서도 매우 중요하며, 개발자가 코드를 쉽게 이해할 수 있어야 메모리 누수나 최적화도 쉽고, 컴퓨터도 잘 이해할 수 있는 코드가 되는것이다

족, 한도 초과, 카드 정지, 시스템 오류 등등)를 가지게 된다. ^{60 61}

그러나 여전히 부족하다. callback과 마찬가지로 promise도 너무 과도하게 사용하면 가독성이 떨어진다. 이를 위해 async/await가 탄생하였다. promise와 비슷하지만 오브젝트 없이 await 키워드를 추가하는 것 만으로도 promise의 기능을 동일하게 수행한다. 이를 통해 좀 더 명확하고 가독성 높은 코드를 설계할 수 있다. await를 사용할 땐 함수를 정의할 때 앞에 키워드로 async를 붙여서 비동기 함수로 세팅해준 다음, 순차적 실행이 필요한 코드에 await를 붙여서 해당 코드는 어떤 동작이 완료될 때까지 대기해야 한다고 알려줘야 한다. ⁶²



```
#CALLBACK
getUser(function(err, user){
  GetProfile(user, function(err, profile){
    GetAccount(profile, function(err, acc){
      GetReport(acc, function(err, report){
        SendStatistics(report, function(e){
          ...
        });
      });
    });
  });
});

#PROMISE
getUser()
  .then(GetProfile)
  .then(GetAccount)
  .then(GetReport)
  .then(SendStatistics)
  .then(function (success) {
    console.log(success)
  })
  .catch(function (e) {
    console.error(e)
  })

#ASYNC/AWAIT
async function SendAsync() {
  let user = await GetUser(1);
  let profile = await GetProfile(user);
  let account = await GetAccount(profile);
  let report = await GetReport(account);

  let send = SendStatistic(report);
  console.log(send)
}
```

63

위의 예시 코드를 보면 callback 은 함수가 끝나면 결과를 포함하여 다시 다음 함수로 순차적으로 호출(콜백)하는 모습을 볼 수 있다. 그러나 매우 복잡하고 가독성도 안좋을 뿐더러 메모리와 최적화 관점에서도 좋지 못하다. 반면 promise 는 가독성은 훨씬 좋아졌지만 여전히 한눈에 명확하게 들어오진 않는다. 그에 반해 await 는 평소 자주 사용하던 변수에 값을 대입함으로써 데이터를 찾기도 편하고, 해당 변수가 어떤 기능을 하는지도 명확하게 구분되어 가독성이 매우 좋아졌다 할 수 있다.

⁶⁰ <https://velog.io/@minew1995/JavaScript-Promise-%EA%B0%9D%EC%B2%B4>

⁶¹ <https://docs.tosspayments.com/blog/using-promises>

⁶² <https://docs.tosspayments.com/blog/async-await-example-%EA%B7%B8%EB%9E%98%EC%84%9C-ayncawait%EB%9E%80>

⁶³ <https://medium.com/@anny.blog/callbacks-vs-promises-vs-async-await-a66668d44c7b>

IV. 웹 개발

앞서 웹 개발에 필요한 언어들에 대해 알아보았다면, 이번엔 실무적인 관점에서 웹 개발이 어떻게 이루어 지는지 중요한 핵심들만 간단히 설명해보겠다. 앞으로 설명할 내용들은 웹 개발 외에서도 사용되는 용어이지만 여기서는 웹 개발에 대한 내용만 설명하겠다.

1. 프론트엔드(front-end)

프론트엔드란 유저에게 시각적으로 보여지는 부분, 즉 페이지(GUI)의 디자인이나 유저와의 인터렉션⁶⁴ 처리를 의미한다. 프론트엔드 개발자는 HTML, CSS, JS(III. 웹 프로그래밍에서 언급된 세가지 언어로, 도서와 해당 목차 내용들은 모두 프론트엔드 관련 내용들이다) 등의 언어를 사용하여 웹을 설계하고 디자인하여 기본적인 정적 웹을 개발한 다음, JS를 이용하여 인터렉션과 백엔드와의 API 통신 및 동적 웹을 설계/개발한다. 유저가 직접적으로 보고 상호작용하는 페이지를 개발하므로 디자인 감각은 물론 접근성과 UX에 대한 이해와 백엔드와의 연결을 위한 서버 및 DB에 대한 지식도 일부 필요하다. 일반적으로 웹 개발자라고 하면 프론트엔드 개발자를 의미할 확률이 더 높다.

⁶⁴ 인터렉션(Interaction)이란 유저와 웹 콘텐츠가 서로 상호작용하여 정보를 주고받는 과정을 의미한다. 본 보고서에서 일반적인 매체와 매체간 상호작용은 '상호작용'으로, 상호작용 중에서 웹 콘텐츠와 유저간의 상호작용은 '인터렉션'으로 구분하여(실제로 업계에서 두 용어는 구분되어 사용된다) 서술하겠다

1) UI(User Interface)

UI는 사용자가 웹 사이트와 인터랙션하는 인터페이스를 의미한다. 웹 페이지의 디자인을 넘어 버튼, 텍스트박스, 메뉴, 하이퍼링크, 아이콘 등등을 포함한다. UI는 사용자가 직접 시각적으로 보게 되며, 깔끔하고 직관적인 UI는 웹에서 중요한 요소이다. 프론트엔드 중에서도 UI를 디자인하는 웹 디자이너(또는 마크업 개발자 또는 웹 퍼블리셔 또는 UI 개발자 등등⁶⁵)는 페이지의 요소들을 배치하고, 크기와 색상을 지정하는 등 웹 사이트의 인테리어를 담당하게 된다.

66

2) UX(User Experience)

UX는 사용자가 웹 사이트를 이용하면서 느끼는 전체적인 경험을 의미한다. UX개발자는 사용자가 웹 사이트를 얼마나 편리하게 이용하는지, 웹 사이트의 목표를 사용자가 얼마나 쉽게 달성하는지, 사용자가 느끼기에 접근성은 어떠한지 등의 실제 유저의 관점에서 편안하게 느끼도록 웹 페이지를 설계하는 역할을 담당한다. 이 중 시각적인 경험인 UI와도 관련이 깊으며 UI와 UX는 동일한 개발자가 설계하거나, 서로 소통하면서 UI를 디자인하여 사용자 경험을 극대화한다. 아무리 유용하고 디자인이 예쁜 웹 사이트여도 사용하기 어렵고 유저에게 친숙하지 못하다면 소용없듯, UX는 실제 유저의 웹 사이트의 평가를 결정짓고 트래픽을 좌우하는 등 웹 개발에서 중요하게 작용한다. UX 개발자는 UX 개선을 위해 유저들의 피드백을 바탕으로 웹을 수정하여 유저들의 요구사항을 최대한 반영하게 된다. ⁶⁷

⁶⁵ 모두 UI를 디자인/설계/개발 하는 사람을 뜻한다

⁶⁶ [사용자 인터페이스, 위키피디아](#)

⁶⁷ <https://darda.net/ui-ux/>

2. 백엔드(back-end)

프론트엔드가 페이지를 구성한다면, 백엔드는 실제 내용(콘텐츠, 기능⁶⁸)을 의미한다. 백엔드 개발자는 웹 사이트의 실질적인 기능을 개발하여 웹 페이지가 의도된 목표를 수행할 수 있도록 해준다. 이때 이 기능 개발에 PHP나 JS⁶⁹, 파이썬 등의 언어가 사용되며, 이 스크립트 및 웹 사이트가 정상적으로 작동할 수 있도록 서버와 데이터베이스(이하 DB)를 구축하고 관리하게 된다. 이를 위해 백엔드 개발자는 백엔드 언어와 서버 및 SQL 그리고 API에 대한 이해를 필요로 한다.^{70 71}

프론트엔드가 돌아가는 프론트엔드 서버에서는 기본적으로 웹 사이트에 관련된 html, css, js 코드가 들어있고, 백엔드 서버에선 DB를 관리하는 SQL이나 다양한 기능을 수행하는 백엔드 코드가 들어있다. 이 두 서버간의 통신은 API의 호출로써 이루어진다. 굳이 프론트엔드 서버에 백엔드 코드까지 넣어도 되지만 따로 분리하여 API를 통해 통신하는 이유는 보안과 편의성 때문이다. 백엔드에는 주로 외부에 노출되면 안되는 요소⁷²들이 포함되어 있어서 외부에 노출되는 프론트엔드 서버는 적절치 못하며, 프론트엔드와 백엔드가 구분되어 있지 않으면 개발하는데 많은 어려움⁷³이 생긴다.

⁶⁸ 웹 페이지의 존재 이유. 구글의 경우 검색, 쿠팡의 경우 제품 구매 등 사이트의 목표(목적)은 분명하며, 백엔드 개발자는 이 목적 달성을 위한 '기능'을 개발한다. 본 보고서에서는 '기능'이라는 단어로 통일하여 사용하겠다

⁶⁹ 프론트엔드의 JS는 동적 페이지 구동이 목적이라면, 백엔드의 JS는 기능 구동이 목적이다

⁷⁰ [프론트엔드와 백엔드, 위키피디아](#)

⁷¹

<https://www.codestates.com/blog/content/%ED%94%84%EB%A1%A0%ED%8A%B8%EC%97%94%EB%93%9C-%EB%B0%B1%EC%97%94%EB%93%9C-%EA%B0%9C%EB%B0%9C%EC%9E%90-%EC%B0%A8%EC%9D%B4%EC%A0%90>

⁷² 주로 API 키나 외부로 유출되면 안되는 코드들

⁷³ 서로 다른 기능의 코드들을 하나로 묶어 놓고 그걸 여러명이 동시에 서로 다른 부분을 수정한다고 생각해보라

프론트엔드와 백엔드의 차이를 이해하기 위해 앞서 들었던 사과 구매를 예시로 들어보자. 온라인 쇼핑몰 사이트의 디자인이나 인터페이스는 UI디자이너가 개발할 것이다. 유저는 이 UI를 통해서 원하는 상품을 쉽게 편하게 볼 수 있다. 다양한 상품들에 대한 정보나 고객의 정보들은 백엔드 개발자가 DB를 통해서 관리할 것이다. 또한 이 데이터들을 가져와서 고객에게 보여주기 위하여 JS를 이용하여 검색 기능을 개발하고, 이를 프론트엔드 개발자가 UI에 존재하는 검색 버튼에 검색 기능을 담당하는 API를 호출하여 상품 정보를 가져오도록 설계한다. 고객은 검색 기능을 통해 사과를 찾아서 구매버튼을 누를 것이다. 이때 프론트엔드는 사용자의 정보들을 통합하여 백엔드의 결제 시스템 API를 호출할 것이고, 백엔드의 결제 시스템에서는 고객의 정보(카드 등)을 토대로 결제를 진행한 다음, 성공 여부를 다시 프론트엔드에 반환하게 될 것이다. 그리고 이 모든 것이 고객이 편리하게 이용할 수 있도록 설계하는 것이 UX 디자이너의 몫이다.

V. 적용

지금까지 설명한 내용들을 활용하여 실제 웹사이트 개발을 어떤 과정을 통해 이루어 졌는지 설명하겠다. 의도된 바는 아니지만 평소처럼 본 프로젝트 또한 애자일하게 진행되었다. 계획이 없으므로 개발 과정이 다소 개연성이 없다고 느껴질 수 있으나 양해바란다. 최대한 계획이 변경된 '사고 과정'을 상세하게 표현하기 위해 시간의 흐름에 따라 간단하게 설명하겠다. 웹 사이트 특성상 결과물을 쉽게 보고 사용해볼 수 있으므로 웹 사이트에서는 볼 수 없는 요소 위주로 설명하겠다.

1. 가사 피피티 메이커

서론에서 언급했듯 교회에서 사용할 찬양 가사를 슬라이드(이하 가사 피피티)로 표시하는 웹 사이트를 개발할 것이다.

먼저 파이썬을 이용하여 만들고자 하는 가사 피피티 프로그램이 실현 가능한지 확인해보았다. Presentation⁷⁴, YouTubeTranscriptApi⁷⁵, LRCLib⁷⁶, request⁷⁷ 등의 라이브러리를 사용하여 '유튜브 영상 링크', '멜론 링크', '음원의 제목과 아티스트 명' 중 하나를 입력하면 스스로 검색하여 가사를 가져오고 이를 PPT로 자동 변환해주는 [코드](#)⁷⁸가 완성되었다. [여기](#)⁷⁹를 클릭하면 해당 코드로 출력된 Rose의 APT.⁸⁰ 곡의 가사 피피티를 볼 수 있다.

⁷⁴ PPT 슬라이드 작업을 위한 라이브러리

⁷⁵ 유튜브의 영상의 자막을 가져오기 위한 라이브러리

⁷⁶ 음원의 가사를 가져오는 API의 라이브러리

⁷⁷ 웹 요청을 보내고 응답 받는 라이브러리

⁷⁸ https://github.com/jjy0809/lrc2ppt/blob/main/gen_lrc.py

⁷⁹ <https://www.lrc2ppt.kr/ppt/?query=apt1>

⁸⁰ 해당 곡을 선택한 이유는 1. 한국어와 영어가 고루 섞여 있으며, 2. 이거 만들 때만 해도 APT.가 멜론 1위였다(나중에 보고서 쓸 때 설명과 이해를 용이하게 하기 위해 인기 있는 곡을 선정했는데 시간이 너무 지나서...)

해당 코드를 사용하여 웹 사이트를 개발하기 위해 파이썬에서 웹 사이트를 구동하기 위한 웹 프레임워크인 Django나 flask 등을 알아보았다. 그러나 생각보다 복잡하고 여러 사유로 인하여 파이썬 대신 일반적인 HTML을 이용하여 개발하기로 결정하였다.

JS로 작동하는 웹 환경에서는 파이썬 코드를 실행할 수 없다. 따라서 해당 파이썬 코드를 JS로 구동시키기 위해 pptgenjs 라이브러리 등을 사용하여 동일한 기능을 수행하는 코드를 작성하였다. 그러나 웹 환경에서는 CORS 정책으로 인해 유튜브 자막 추출 기능을 사용할 수 없다. 어쩔 수 없이 해당 기능은 제거하고 최종적으로는 가사를 검색하는 기능만 남겨두었다. 이 가사 검색에는 LRCLIB과 Musixmatch API를 사용하였다.⁸¹

처음에는 모든 JS 코드를 프론트에서 실행시켰다. 그러나 보안등의 이유로 가사를 검색하는 코드와 PPT를 생성하는 코드 중 일부를 백엔드 서버로 옮기기 위해 AWS의 Lamda를 사용하여 API를 구축하고 프론트와 백을 API를 통하여 통신하도록 설계하였다. Lambda에서는 node.js를 통해 js 코드를 작동하여 결과물은 다시 웹에 전송시켜 유저가 받아볼 수 있도록 한다.

전체적인 UI를 HTML/CSS를 통해 만들고 디자인을 UX 개선을 위해 여러 차례 수정하고, 도중에 발견된 여러 문제점들과 JS 코드 수정 등의 과정⁸²을 통해 최종적인 웹 사이트를 개발할 수 있었다.

전체적으로 디자인을 깔끔하고 직관적으로 보일 수 있도록 노력하였다. 또한 모바일에서도 사용할 수 있도록 반응형 가변 너비(상대 수치)를 사용하여 모든 크기의 화면에서 페이지가 정상적으로 표시되도록 만들었다.

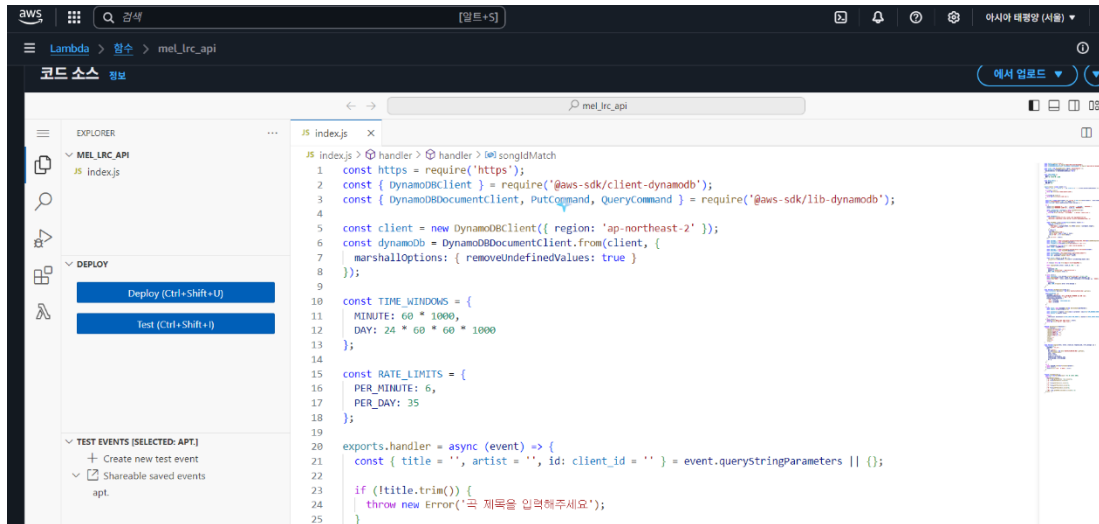
또한 사용법이 익숙치 않을 유저들을 위해 사진을 포함한 사용 설명서를 제작하여 사용법을 안내하고 있다.

완성된 사이트는 <https://www.lrc2ppt.kr/> 에서 볼 수 있다.

⁸¹ 초기 웹 사이트의 모습은 <https://lrc2ppt.kr/main/> 에서 볼 수 있다

⁸² 너무 많아서 생략한다. 수시로 백업해둔 소스 파일들은 <https://github.com/jjy0809/lrc2ppt> 에서 볼 수 있다(index, style, script 뒤의 (n)에서 n 이 더 낮을수록 초기 버전이다)

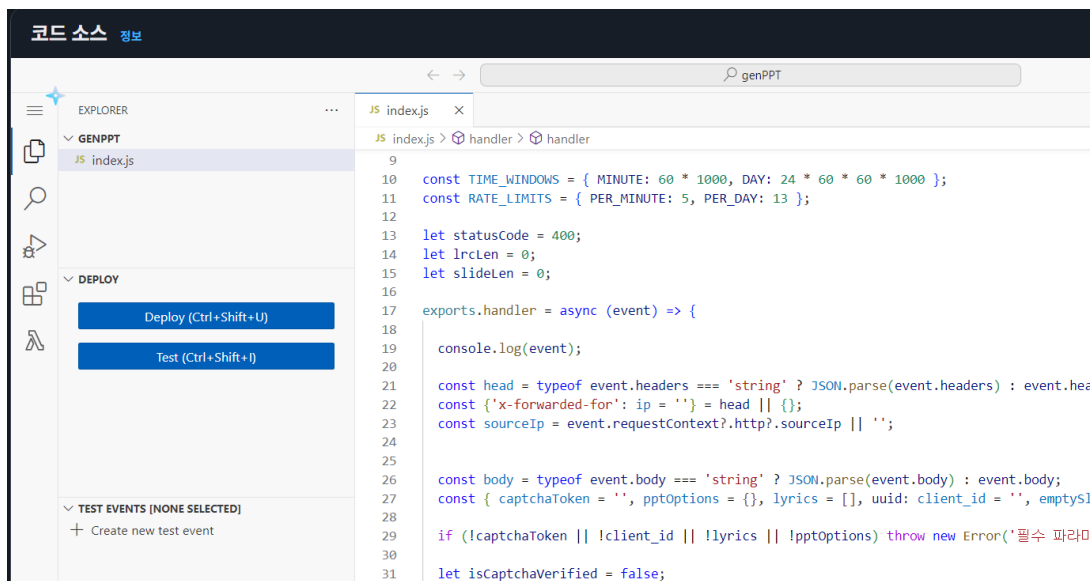
아래는 백엔드 API 코드의 일부이다.



The screenshot shows the AWS Lambda console for the function 'mel_lrc_api'. The code is written in JavaScript and is located in the 'index.js' file. The code defines a handler function that takes an event object as input. It uses the AWS SDK for JavaScript to interact with DynamoDB. The code includes constants for time windows and rate limits. The handler function checks if the event has a 'client_id' parameter. If it does, it queries the DynamoDB database for the corresponding record. If the record is found, it returns the record. If the record is not found, it throws an error.

```
1 const https = require('https');
2 const { DynamoDBClient } = require('@aws-sdk/client-dynamodb');
3 const { DynamoDBDocumentClient, PutCommand, QueryCommand } = require('@aws-sdk/lib-dynamodb');
4
5 const client = new DynamoDBClient({ region: 'ap-northeast-2' });
6 const dynamoDb = DynamoDBDocumentClient.from(client, {
7   marshalOptions: { removeUndefinedValues: true }
8 });
9
10 const TIME_WINDOWS = {
11   MINUTE: 60 * 1000,
12   DAY: 24 * 60 * 60 * 1000
13 };
14
15 const RATE_LIMITS = {
16   PER_MINUTE: 6,
17   PER_DAY: 35
18 };
19
20 exports.handler = async (event) => {
21   const { title = '', artist = '', id: client_id = '' } = event.queryStringParameters || {};
22
23   if (!title.trim()) {
24     throw new Error('곡 제목을 입력해주세요!');
25   }
26 }
```

- 가사 검색 API



The screenshot shows the AWS Lambda console for the function 'genPPT'. The code is written in JavaScript and is located in the 'index.js' file. The code defines a handler function that takes an event object as input. It uses the AWS SDK for JavaScript to interact with DynamoDB. The code includes constants for time windows and rate limits. The handler function checks if the event has a 'client_id' parameter. If it does, it queries the DynamoDB database for the corresponding record. If the record is found, it returns the record. If the record is not found, it throws an error.

```
9
10 const TIME_WINDOWS = { MINUTE: 60 * 1000, DAY: 24 * 60 * 60 * 1000 };
11 const RATE_LIMITS = { PER_MINUTE: 5, PER_DAY: 13 };
12
13 let statusCode = 400;
14 let lrcLen = 0;
15 let slideLen = 0;
16
17 exports.handler = async (event) => {
18   console.log(event);
19
20   const head = typeof event.headers === 'string' ? JSON.parse(event.headers) : event.headers;
21   const { 'x-forwarded-for': ip = '' } = head || {};
22   const sourceIp = event.requestContext?.http?.sourceIp || '';
23
24   const body = typeof event.body === 'string' ? JSON.parse(event.body) : event.body;
25   const { captchaToken = '', pptOptions = {}, lyrics = [], uuid: client_id = '', emptySl
26
27   if (!captchaToken || !client_id || !lyrics || !pptOptions) throw new Error('필수 파라미
28
29   let isCaptchaVerified = false;
30
31 }
```

- PPT 생성 API

아래는 API 사용 로그 중 일부이다

테이블: **trc_scr - 반환된 항목 (400)**
 소문 시작 날짜: 6월 29, 2025, 17:34:28

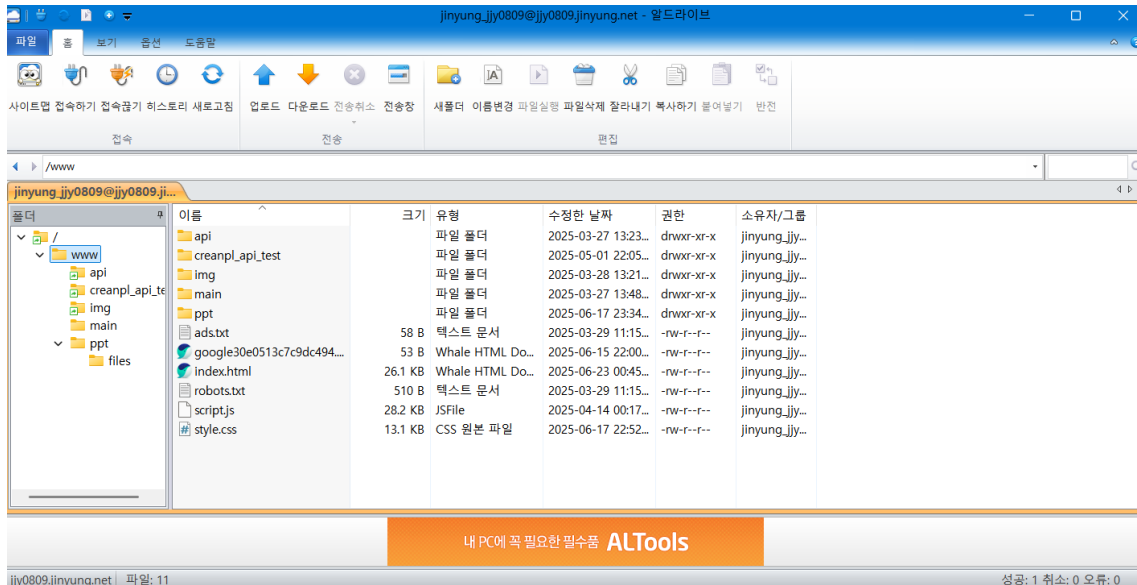
<input type="checkbox"/>	IP (문자열) ▾	ID (숫자) ▾	API ▾	artist ▾	client_id ▾	error_message ▾	response_code ▾	time ▾	title
<input type="checkbox"/>	110.8.56.11	15066967981	search	<empty>	307d9544-...	<empty>	200	20250624 ...	먼저 그 나라와
<input type="checkbox"/>	110.8.56.11	15066950304	search	<empty>	307d9544-...	가사를 찾을 수 없...	400	20250624 ...	먼저 그 나라와 의를 구하라
<input type="checkbox"/>	110.8.56.11	15066542455	search	<empty>	307d9544-...	<empty>	200	20250624 ...	나를 지으신 이가 하나님
<input type="checkbox"/>	110.8.56.11	15066235444	search	<empty>	307d9544-...	<empty>	200	20250624 ...	허락하신 새 땅에
<input type="checkbox"/>	218.149.1...	14913515285	search	<empty>	85e61636-...	음원을 찾을 수 없...	400	20250622 ...	유리와예배
<input type="checkbox"/>	211.243.8...	14886374614	search	<empty>	824da427-4...	<empty>	200	20250622 ...	말할 수 없네
<input type="checkbox"/>	220.75.20...	14880333634	search	<empty>	62629729-...	<empty>	200	20250622 ...	주의 자비가 내려와
<input type="checkbox"/>	220.75.20...	14878421519	search	<empty>	62629729-...	<empty>	200	20250622 ...	사명
<input type="checkbox"/>	220.75.20...	14878413363	search	<empty>	62629729-...	음원을 찾을 수 없...	400	20250622 ...	주 자비 충주게 하네
<input type="checkbox"/>	222.103.1...	14875564923	search	<empty>	0273f27a-5...	<empty>	200	20250622 ...	내 마음의 한 자리
<input type="checkbox"/>	222.103.1...	14875350206	search	<empty>	0273f27a-5...	<empty>	200	20250622 ...	나로부터 시작되리
<input type="checkbox"/>	58.237.12...	14395215309	search	<empty>	2b0d5e45-...	<empty>	200	20250616 ...	임페
<input type="checkbox"/>	58.237.12...	14395196141	search	<empty>	2b0d5e45-...	<empty>	200	20250616 ...	꽃들도
<input type="checkbox"/>	119.197.1...	14276129776	search	<empty>	f6d2c2b5-9...	<empty>	200	20250615 ...	만복의근원하나님
<input type="checkbox"/>	14.46.177...	14273658851	search	<empty>	0273f27a-5...	<empty>	200	20250615 ...	내마음의 할자리
<input type="checkbox"/>	14.46.177...	14273645781	search	<empty>	0273f27a-5...	음원을 찾을 수 없...	400	20250615 ...	우리들을 위하여
<input type="checkbox"/>	14.46.177...	14272069139	search	<empty>	0273f27a-5...	<empty>	200	20250615 ...	일어나의 주의 복성

- 가사 검색 API 로그

<input type="checkbox"/>	IP (문자열) ▾	ID (숫자) ▾	client_id ▾	error_message ▾	isCaptchaVerified ▾	lyrics_length ▾	pptOptions ▾	response_code ▾	slide_length ▾	time
<input type="checkbox"/>	110.8.56.11	15431363119	307d9544-...	<empty>	true	20	{ "lineMode" ...	200	22	20250622 ...
<input type="checkbox"/>	110.8.56.11	15431171949	307d9544-...	<empty>	true	50	{ "lineMode" ...	200	48	20250628 ...
<input type="checkbox"/>	211.97.11...	15390071295	9886c36a-a...	<empty>	true	62	{ "lineMode" ...	200	38	20250628 ...
<input type="checkbox"/>	121.135.9...	15228648338	fbe57a28-7...	<empty>	true	39	{ "lineMode" ...	200	24	20250626 ...
<input type="checkbox"/>	121.135.9...	15228246270	fbe57a28-7...	<empty>	true	2	{ "lineMode" ...	200	1	20250626 ...
<input type="checkbox"/>	211.234.1...	15066688486	3fab5aae-c...	<empty>	true	10	{ "lineMode" ...	200	6	20250624 ...
<input type="checkbox"/>	110.8.56.11	15066582260	307d9544-...	<empty>	true	20	{ "lineMode" ...	200	19	20250624 ...
<input type="checkbox"/>	110.8.56.11	15066278937	307d9544-...	<empty>	true	16	{ "lineMode" ...	200	16	20250624 ...
<input type="checkbox"/>	220.75.20...	14880356174	62629729-...	<empty>	true	25	{ "lineMode" ...	200	18	20250622 ...
<input type="checkbox"/>	220.75.20...	14878322209	62629729-...	<empty>	true	13	{ "lineMode" ...	200	8	20250622 ...
<input type="checkbox"/>	222.103.1...	14875631457	0273f27a-5...	<empty>	true	95	{ "lineMode" ...	200	43	20250622 ...
<input type="checkbox"/>	175.198.2...	14561765367	cb909aee-c...	<empty>	true	45	{ "lineMode" ...	200	38	20250618 ...
<input type="checkbox"/>	175.198.2...	14561512972	cb909aee-c...	<empty>	true	42	{ "lineMode" ...	200	33	20250618 ...
<input type="checkbox"/>	175.198.2...	14561343620	cb909aee-c...	<empty>	true	36	{ "lineMode" ...	200	29	20250618 ...
<input type="checkbox"/>	175.198.2...	14561100553	cb909aee-c...	<empty>	true	45	{ "lineMode" ...	200	38	20250618 ...
<input type="checkbox"/>	58.237.12...	14395261427	2b0d5e45-...	<empty>	true	119	{ "lineMode" ...	200	86	20250616 ...
<input type="checkbox"/>	112.153.2...	14296056550	2681b0ec-c...	<empty>	true	5	{ "lineMode" ...	200	3	20250615 ...
<input type="checkbox"/>	221.146.2...	14291687691	ad6f4995-e...	<empty>	true	88	{ "lineMode" ...	200	74	20250615 ...

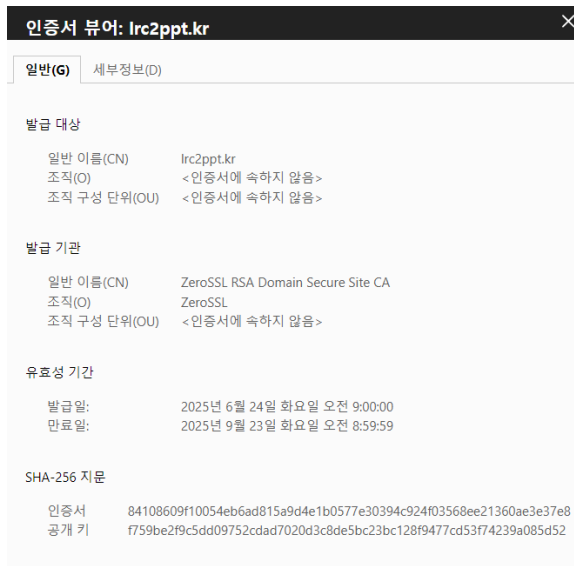
- PPT 생성 API 로그

아래는 웹 호스팅을 위해 사용한 프론트엔드 서버의 FTP 프로그램이다



- 프론트엔드 서버⁸³

또한 웹 사이트의 안전성과 신뢰를 위해 SSL 인증서를 추가하여 https 연결이 가능하도록 만들었다

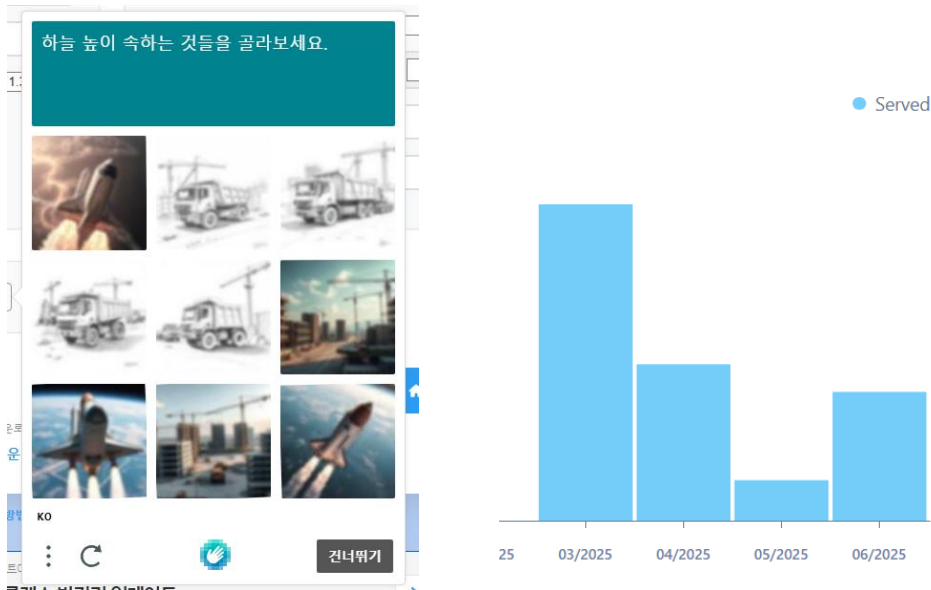


- SSL 인증서 정보

⁸³ 파일을 업로드하여서 소스코드를 추가할 수 있다. 서버의 웹서버 프로그램은 nginx이다

1) 캡차(Captcha)

자동화 프로그램에 의한 과도한 사용이나 공격을 막기 위해 캡차를 삽입하여 결과물인 PPT를 다운로드 하려면 캡차를 먼저 풀어야 하도록 설정하였다. 유명한 액티브 캡차⁸⁴로는 reCaptcha와 hCaptcha가 있다. 여기서는 비용이 저렴한 hCaptcha를 사용하였다.



좌측 사진은 사이트에 hCaptcha가 적용된 모습이다. PPT 파일을 다운로드 하기 위해서는 해당 캡차 퍼즐을 풀어야만 한다. 우측 사진은 캡차 사용량이다.⁸⁵

⁸⁴ 캡차는 사람이 직접 퍼즐을 풀어서 로봇이 아님을 증명하는 active 캡차와 인공지능이 패턴을 분석하여 사람인지 아닌지 판단하는 passive 캡차로 나뉘어진다. 최근에는 두 방법을 섞어서 사용하는 경향이 있지만, 여기서는 캡차를 보여주기 위해 active만을 사용하겠다

⁸⁵ 캡차를 3월에 도입하여 테스트를 하느라 다른 달에 비해서 사용량이 많이 나온다

2) 도메인

앞서 3) 도메인 네임에서 설명했듯이 사이트의 도메인은 중요하다. 사이트의 정체성을 함축하는 상징적인 도메인네임을 선정하는 동시에, 외우기 쉽고 간결해야 한다. 도메인 네임은 가사를 피피티로 변환한다는 의미의 lrc2ppt⁸⁶로 선정하였고, TLD는 신뢰도 높으면서 한국 내에서 한국인을 대상으로 서비스하는 사이트의 특성을 살려 kr로 정하여 최종적인 도메인은 lrc2ppt.kr로 설정했다. 아래는 도메인 등록확인서이다.

HOSTING.KR

도메인 등록 확인서 (Certificate Domain Proxy Registration)

도메인	lrc2ppt.kr
등록일	2025-03-27
만료일	2026-03-27
소유자명	정 준영 / Jeong JunYoung
소유자 전화번호	+82 01086 [REDACTED]
관리자명	정 준영 / Jeong JunYoung
관리자 이메일	happy [REDACTED] naver.com
네임서버 (호스트)	ns.jinyung.net ns1.jinyung.net ns2.jinyung.net ns3.jinyung.net
등록 대행자 (Registrar)	메가존(주)

당사에 다음과 같이 도메인이 등록되었음을 확인합니다.

발급일시 : 2025-06-29

메가존(주)



네임서버(정확히는 신뢰가능한 네임서버)는 프론트엔드 서버 업체의 네임서버를 이용하였다.

⁸⁶ lrc는 가사(Lyrics)의 약자, 2는 to의 약자로 주로 변환기에 많이 사용된다(ex png to jpg)

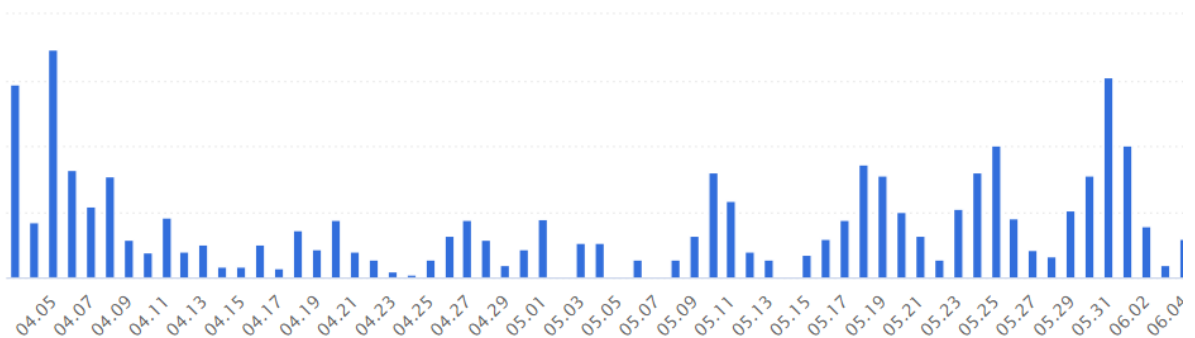
3) 파비콘(favicon)

도메인 만큼 중요한게 사이트의 파비콘이다. 구글의 G나 네이버의 N, 카카오톡의 말풍선만 보아도 어떤 브랜드인지 알 수 있듯 파비콘을 통해 브랜드 이미지를 나타낼 수 있다. 곡의 가사를 표시하는 슬라이드를 표현하기 위해 직접 파비콘을 간단하게 디자인하여 적용하였다.

파비콘은 <https://www.lrc2ppt.kr/img/favicon.png> 에서 볼 수 있다.

4) 광고

프론트 서버 호스팅 비용⁸⁷과 백엔드용 AWS 람다 비용, 캡차 비용⁸⁸, 도메인 비용 등 사이트 운영에 들어가는 비용이 꽤 발생하자, 장기적인 사이트 운영을 위해 비영리로 운영되는 사이트이지만 광고를 삽입하여 비용을 어느정도 충당하려고 한다. 직접 광고주를 구해서 광고를 게재하는 것은 불가능하니 대행 업체를 이용하여 광고를 게재하였다. 대표적인 대행 업체로는 구글 애드센스와 카카오 애드핏이 있으며, 두 업체 모두 심사를 통해서 사이트가 광고를 게재할 만한지 검증한 뒤 승인이 되어야 광고를 게재할 수 있다. 구글은 심사가 많이 까다롭고 오래걸리는 관계로 카카오 애드핏 서비스를 채택하여 심사를 받았으며, 승인받았다. 광고는 최대한 콘텐츠를 가리지 않도록 좌우 또는 상하에 배치하였으며, UX에 최대한 해를 끼치지 않는 선에서 광고 수익이 적절히 발생하도록 하였다. 아래는 광고수익 그래프이다.



광고는 어떤 광고가 표시되냐에 따라 단가가 달라지므로 단순히 조회수만이 아닌 클릭수와 광고 단가도 수익에 큰 영향을 미친다. 전체적으로 그래프가 금~일요일에 솟아 올랐다가 떨어졌다가를 반복하는 모습이다.

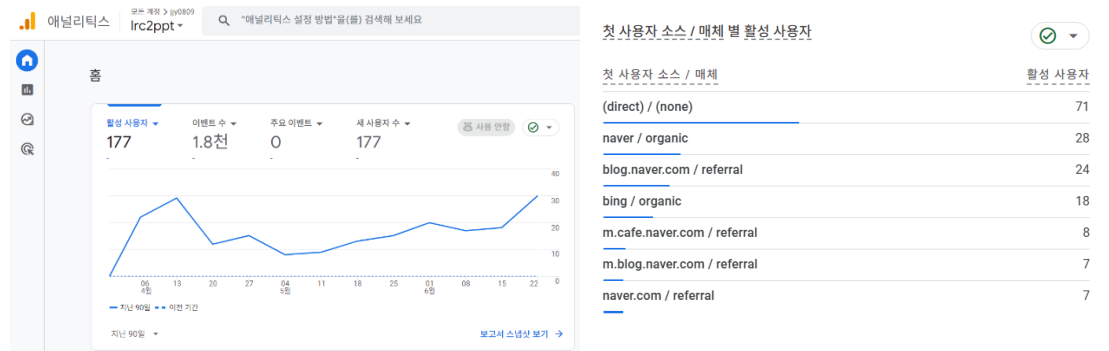
광고 수익은 평일 기준 하루에 몇십원, 주말 기준 몇십~몇백원 수준으로 상당히 많이 짜다. 위 그래프에서 가장 높은 축이 320원 정도로 일주일 수익이 500원 넘기도 힘들다. 애초에 비용 충당이 목적이므로 이 정도에 만족하자.

⁸⁷ 지인이 호스팅 업체를 운영하여 도움을 받아 거의 무료로 서버를 사용할 수 있었다. 도움이 없었다면 사이트 운영 비용이 상당하여 유료 시스템이나 후원을 받아야 했을 것이다

⁸⁸ AWS와 캡차 모두 일정량의 트래픽은 무료이지만 살짝 넘어서서 과금이 진행중이다...

5) 구글 애널리틱스(Google Analytics)

사이트 트래픽을 분석하고 관리하기 위해 구글의 마케팅 서비스 중 하나인 애널리틱스를 사용하여 유입 경로와 트래픽 추이 등을 분석해보았다.



- 사이트 트래픽(좌) / 유입 경로⁸⁹(우)

시/군/구▼ 별 활성 사용자▼

시/군/구	활성 사용자
Seoul	39
Daegu	35
Busan	10
Incheon	5
Anseong-si	4
Gwangju	4
Hwaseong-si	4

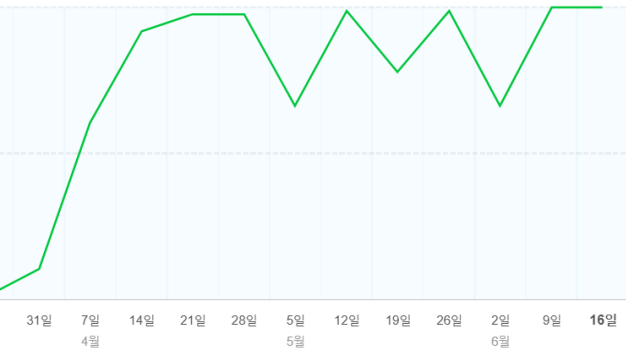
- 트래픽 지역

⁸⁹ 사이트에 도달하게 된 경로. direct의 경우 바로 url을 검색하여 들어온 경우, naver의 경우 naver 검색을 통해, naver blog의 경우 블로그를 통해서 유입된 경우이다

2. 홍보

1) 블로그

이왕 개발한 김에 많은 사람들이 사용하면 좋을 것 같아 사람들에게 해당 사이트를 알릴 방법을 고민하던 중, 사람들이 블로그를 통해 가사 피피티를 많이 찾는다는 사실을 알게되고 블로그를 개설해서 사이트를 홍보하자는 계획을 세웠다. 대표적으로 인기 있는 찬양들을 선정하여 '가사 피피티 메이커'를 이용하여 자동으로 PPT를 생성한 뒤, 직접 블로그 포스팅 작업을 해줬다. 블로그는 <https://blog.naver.com/lrc2ppt> 에서 볼 수 있다.



블로그 조회수이다. 매주 약 100회 정도의 조회수를 기록하고 있으며, 사이트 광고 수익과 비슷하게 토~일요일에 조회수가 각각 약 40회 정도로 조회수의 대다수를 차지하고 있다.

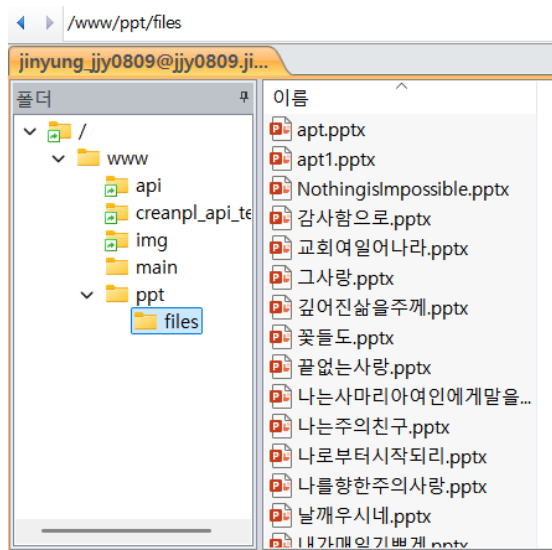
조회수 순위 2025.05. 월 일간 주간 월간

게시물		주제		
순위	제목	조회수	타입	작성일
1	꽃등드 - 교회 찬양 CCM 가사 PPT 피피티 공유	28	글	2025.04.06 (일)
2	주를 찾는 모든 자들이 - 교회 찬양 CCM 가사 PPT 피피티...	16	글	2025.04.15 (화)
3	하나님의 부르심 - 교회 찬양 CCM 가사 PPT 피피티 공유	13	글	2025.04.06 (일)
3	여호와께 돌아가자 - 교회 찬양 CCM 가사 PPT 피피티 ...	13	글	2025.04.15 (화)
5	원하고 바라고 기도합니다 - 교회 찬양 CCM 가사 PPT ...	12	글	2025.04.15 (화)
6	우리 살아가는 모든 날들이 - 교회 찬양 CCM 가사 PPT ...	11	글	2025.04.06 (일)
6	우린 주를 만나고 - 교회 찬양 CCM 가사 PPT 피피티 공유	11	글	2025.04.15 (화)
8	주님의 선하심 - 교회 찬양 CCM 가사 PPT 피피티 공유	10	글	2025.04.06 (일)
8	오늘 이곳에 계신 성령님 - 교회 찬양 CCM 가사 PPT 피...	10	글	2025.04.15 (화)
8	다시 일어나 - 교회 찬양 CCM 가사 PPT 피피티 공유	10	글	2025.04.15 (화)
11	시편 139편 - 교회 찬양 CCM 가사 PPT 피피티 공유	9	글	2025.04.06 (일)
11	주와 같이 갈 가는것 - 교회 찬양 CCM 가사 PPT 피피티 ...	9	글	2025.04.06 (일)
11	길어진 삶을 주께 - 교회 찬양 CCM 가사 PPT 피피티 공유	9	글	2025.04.06 (일)

- 5월 글 조회수 순위

2) 가사 피피티 파인더

블로그에서 사이트 접속률이 조회수에 비해 저조하자, 직접 PPT파일을 블로그 글에 업로드하는 대신, 사이트에서 다운로드 받도록 유도하여 사이트 트래픽을 증가하고 자연스럽게 '가사 피피티 메이커'로 유입되도록 일부 인기있는 글에서 PPT 파일을 삭제하고 '가사 피피티 파인더'의 링크를 걸어두었다. 사이트는 기존의 '가사 피피티 메이커'의 디자인과 코드를 사용하여 새로운 하위 사이트를 만들었고, PPT파일은 프론트 서버에 업로드하여 파인더를 통해 파일을 검색하고 미리보기를 제공하는 동시에 파일을 즉시 다운로드 할 수 있도록 설계하였다. 또한 존재하지 않는 파일은 피피티 메이커에서 직접 만들도록 유도하고 있다. <https://www.lrc2ppt.kr/ppt/> 를 통해서 접속할 수 있으며, 뒤에 쿼리 스트링으로 <https://www.lrc2ppt.kr/ppt/?query=apt1> 처럼 query 파라미터를 통해 바로 해당 곡의 PPT 파일을 볼 수 있도록 설계하여 블로그에서 파인더로 넘어갈 때 다시 곡을 검색하지 않고도 바로 파일을 다운로드 받을 수 있도록 만들었다.



- 프론트서버에 PPT가 업로드된 모습

추가로 초기의 파이썬 코드로 생성한 apt PPT와 새롭게 node.js로 개발하고 업그레이드한 apt PPT를 미리보기로 쉽게 비교할 수 있다.

<https://www.lrc2ppt.kr/ppt/?query=apt> <- '가사 피피티 메이커'로 만든 PPT

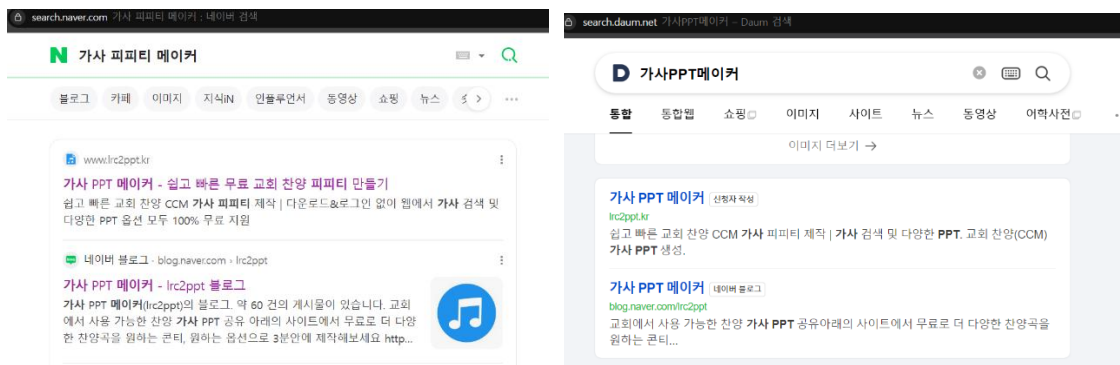
3) 검색

가장 확실한 마케팅은 검색 엔진에 사이트가 노출되는 것이다. 이를 위해 검색 엔진의 크롤링⁹⁰ 봇이 사이트를 수집할 수 있도록 robots.txt⁹¹를 설정해 줬다. 해당 내용은 <https://www.lrc2ppt.kr/robots.txt> 에서 볼 수 있다.⁹²

대한민국에서 가장 많이 사용하는 네이버와 다음에 각각 사이트를 등록하였으며, 연관 검색어를 검색하면 사이트가 노출된다.

https://search.naver.com/search.naver?sm=tab_hy_top&where=nexearch&ssc=tab.nx.all&query=가사PPT메이커

<https://search.daum.net/search?w=tot&DA=JU5&q=가사PPT메이커>



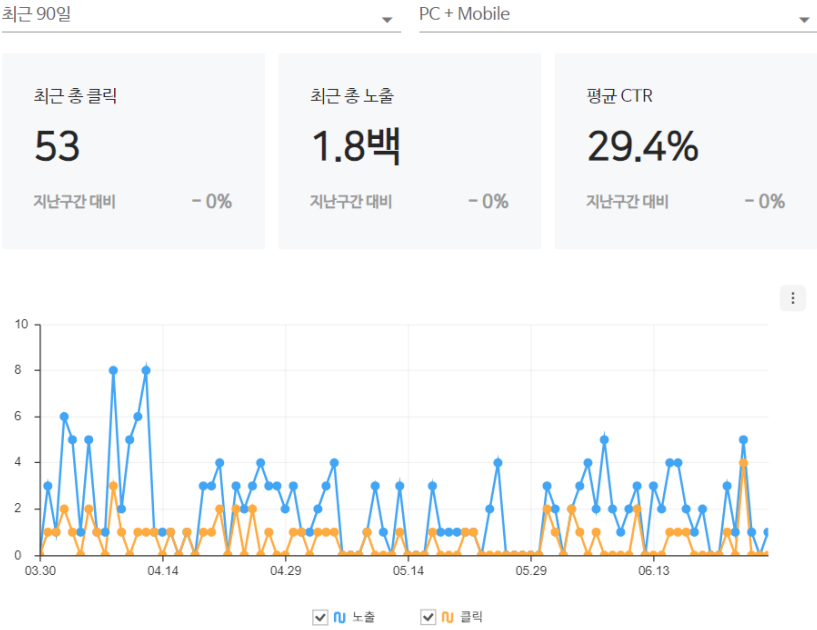
- 네이버 검색 결과(좌) / 다음 검색 결과(우)

⁹⁰ 검색엔진에서 사이트를 표시하기 위해선 검색엔진 DB에 해당 사이트가 존재해야 한다. 이때 새로운 사이트들을 수집하는 것이 크롤링봇의 역할이며, 수집되지 못한 사이트는 고립되어 딥웹이 된다

⁹¹ 사이트를 크롤링하는 봇들의 접근을 허용하거나 막는 규칙. 법적 구속력은 없으나 윤리적으로 해당 규약을 지켜주는 것이 불문율이다

⁹² 네이버 크롤링 봇의 이름은 특이하게 yeti 이다. 네이버와 구글을 포함한 대부분의 봇들은 허용해주고 그 외의 봇들은 와일드카드(*)를 통해 거부해줬다

노출/클릭 현황



- 네이버 검색 노출 및 클릭 통계

No	검색 키워드	클릭	노출	CTR(%)
1	찬양 가사 피피티	10	23	43.5
2	가사 피피티 메이커	5	11	45.5
3	가사 피피티	3	7	42.9
4	교회 가사 피피티	2	7	28.6
5	교회 찬양 가사 ppt	2	3	66.7
6	찬양 피피티 만들기	2	3	66.7
7	교회 찬양 피피티	1	7	14.3
8	교회 찬양 가사 피피티	1	6	16.7
9	찬양피피티 만들기	1	5	20
10	찬양 가사 ppt 만들기	1	4	25

< 1 2 3 >

- 검색 키워드 순위

VI. 결론

실생활에서 불편한 요소를 찾아 이를 해결하기 위하여 직접 자동화 프로그램을 만들고 이를 웹 사이트를 통해 불특정 다수에게 배포하였으며, 광고를 게재하여 지출 비용을 충당하기 위한 수익 모델을 설계하고 실제로 수익을 창출해냈다. 유의미한 트래픽 및 사용량을 달성하여 예상보다 많은 사람들이 내가 직접 개발한 사이트의 콘텐츠를 유용하게 이용해 주었다. 오랜 세월 프로그래밍을 배우고 익힌 노하우와 지식을 바탕으로 지금까지 각종 인터넷 블로그와 책, 학교의 프로그래밍 수업 등 여러곳에서 여러 사람에게 배워온 것의 종합체이자 종지부로서 이번 프로젝트를 진행하였다. 이번 프로젝트를 통해 실제 개발 과정을 실무자의 입장에서 직접적으로 체감하였고 아직 배워야 할 내용들이 많다는 사실 또한 느꼈다. 생각보다 웹 사이트 하나 만드는 것조차 매우 어렵고 많은 오류들과 문제들이 수시로 발생하였지만, 이들을 극복하고 해결하는 과정을 통해 문제 해결력 또한 많이 향상되었으며 문제를 해결해나가며 성취감을 느끼고 코딩이 더욱 재밌어 지며 개발자를 진로로 삼길 잘했다는 생각이 들었다. 기획부터 풀스택⁹³ 개발과 마케팅 및 유지 보수까지 혼자서 전과정을 경험해보며 앞으로 타인들과 함께 팀으로 개발하는 공동 프로젝트에서도 이 경험을 바탕으로 더욱 원활한 팀워크를 발휘할 수 있을것이라 기대한다.

본 프로젝트에 사용된 코드 및 파일 중 보안상 문제되는 파일을 제외한 모든 코드 및 파일은 [깃허브](#)에 업로드 되어 있다. 특히 최종적으로 완성된 결과물인 사이트의 전체 파일은 <https://github.com/jjy0809/lrc2ppt/tree/main/www>에서 볼 수 있다.

- 가사 피피티 메이커 | <https://www.lrc2ppt.kr/>
- 가사 피피티 파인더 | <https://www.lrc2ppt.kr/ppt/>
- 가사 피피티 블로그 | <https://blog.naver.com/lrc2ppt>

⁹³ 프론트엔드와 백엔드를 동시에 개발하는 개발자