**서버(Server) vs. 클라이언트(Client)**  
서비스를 제공하느냐/제공받느냐에 따른 구분. 외부에 필요한 서비스를 제공하도록 만든 컴퓨터나 프로그램 영역을 서버라고 한다면, 여기에 접속해 서비스를 요청하고 제공받는 유저가 클라이언트가 된다.   
  
**프론트엔드 개발자(Front-end Developer)**  
웹 브라우저를 통해 유저가 직접 마주하는 웹 서비스의 앞단(front-end)을 담당하는 개발자. 클라이언트/서버를 기준으로 보면 웹 페이지 화면을 비롯한 클라이언트 영역을 프론트엔드라고 할 수 있다.    
  
**백엔드 개발자(Back-end Developer)**  
프론트엔드 개발자의 반대 개념으로, 웹 서비스의 뒷단(Back-end)을 담당하는 개발자. 주로 유저에게 보이지 않는 DB와 API로 이루어진 서버 영역을 관리, 개발하는 역할을 한다.   
  
**풀스택(Full-Stack)**  
프론트엔드 개발과 백엔드 개발 영역을 통틀어 이르는 말, 혹은 모두 다룰 수 있는 개발자. (‘풀스택 개발자’) 희귀종, 유니콘, 일당백이 될 수도 있지만 쏟아지는 업무에 시달리거나 이도저도 아닌 빈 스택(Empty Stack)이 될 수도 있으니 너무 막연한 환상은 금물.   
  
**프레임워크(Framework)**  
프로그램을 개발하기 위한 구조를 제공하는 개발 환경. 코딩을 할 때 자주 쓰이는 여러 클래스(Class, 변수와 함수의 묶음)를 프레임워크가 정해둔 흐름에 맞춰 쓸 수 있도록 지원해 주기 때문에 복잡하거나 반복되는 작업에 대한 부담을 덜어줄 뿐만 아니라 개발에 들이는 시간 대비 프로그램의 질을 높일 수 있다. (자바스크립트에서는 뷰(Vue.js)와 앵귤러(Angular)가, 자바에서는 스프링(Spring)이 대표적인 웹 프레임워크로 꼽힌다.)   
  
**라이브러리(Library)**  
프로그램을 개발하는 데 필요한 여러 기능을 활용할 수 있도록 묶어놓은 함수 또는 기능의 집합. 프레임워크가 개발을 제어하는 틀거리를 제공해준다면, 라이브러리는 도서관에서 책을 빌리듯 개발자가 필요한 기능을 마음대로 가져다 쓸 수 있는 일종의 모음집에 가깝다. 즉 정해진 방식으로 개발을 하게끔 정해주는 프레임워크와 달리 제어권이 개발자에게 주어지는 셈. (자바스크립트 웹 개발에 쓰이는 리액트(React)와 제이쿼리(jQuery)가 바로 라이브러리다.)   
  
**플러그인(Plug-in)**  
어떤 특정한 기능을 해결하는 데 쓸 수 있도록 미리 만들어 놓은 코드 및 데이터의 모음. 한 라이브러리나 프레임워크에서 여러 개의 플러그인을 제공한다고 볼 수 있다.   
  
**웹 퍼블리셔(Web Publisher)**  
디자인 시안에 맞춰 웹 화면을 구현하는 웹 퍼블리싱(Web Publishing)을 담당하는 직무로 한국에서만 쓰이는 말이다. 프론트엔드 개발에서 웹 화면과 관련한 시각적인 면을 담당하는 개념을 떼놓은 것. 디자인은 물론 웹 표준과 호환성을 고려해 코딩을 진행하고, 디자이너와 프론트엔드 개발자 사이에서 의견을 조율하기도 한다. 주로 HTML과 CSS을 사용하며 자바스크립트 기반의 제이쿼리(jQuery), 부트스트랩(Bootstrap) 같은 라이브러리나 프레임워크까지 사용하는 편. 마크업 언어인 HTML, CSS를 다루기 때문에 마크업 개발자(Markup Developer)라고도 한다.  
같은 퍼블리셔라고 해도 회사마다, 담당하는 사람마다 맡은 일의 범위가 제각기 다르다. 어떤 회사에서는 프론트엔드 개발자가 퍼블리싱까지 담당하지만, 웹 퍼블리셔를 따로 두거나 디자이너가 웹 퍼블리싱까지 맡는 경우도 많다.    
  
**마크업 언어(Markup Language)**  
태그 등을 이용해 문서나 데이터의 구조를 표시한 언어. 프로그래밍 언어와는 다르며, HTML, CSS, XML이 대표적인 마크업 언어다. (빛보다 빠르게 개발자를 소환하고 싶다면 “HTML은 프로그래밍 언어”라고 말해보시길!)   
  
**웹 접근성(Web Accessibility)**  
누구에게나 평등한 웹 환경을 구성하는 요소. 웹 표준을 지킴으로써 장애를 가진 사람, 고령자, 어린이 등이 다른 유저들과 마찬가지로 웹에서 제공하는 정보를 원활하게 이용할 수 있도록 보장하는 것. (시각장애인에게 시각 콘텐츠에 대한 의미나 용도를 이해할 수 있도록 대체 텍스트를 제공하거나, 키보드만으로 서비스의 모든 기능을 사용할 수 있도록 하는 것 등이 기준에 포함된다.)   
  
**검색 엔진 최적화(Search Engine Optimize)**  
줄여서 SEO. 웹 페이지를 구글 등 검색 엔진이나 포털사이트의 기준에 맞춰 구성하는 작업. 사람들이 많이 검색할 만한 관련 키워드를 배치하고, 웹 표준을 지키며, 스팸으로 분류되지 않게끔 하는 등 웹 페이지 링크가 더 많이 노출되도록 처리한다. 온라인 마케팅의 필수 요소로도 꼽히는데, SEO가 잘 적용된 웹 페이지는 검색 시 상단에 오를 수 있기 때문.   
  
**버그(Bug)**  
소프트웨어에서 발생하는 예견치 못한 오류나 오작동.  
  
**에러(Error)**  
유저가 입력한 내용이 잘못됐을 때 발생하는 문제. 잘못 짠 코드처럼 내부 문제로 예상하지 못한 문제가 일어나는 버그와 달리, 외부에 있는 유저가 오탈자를 내거나 띄어쓰기를 실수하는 등 내부에서 정한 형식을 따르지 않을 때 주로 발생한다. (예를 들어 숫자만 적도록 되어있는 주민번호 입력창에 inflearn-newbie라고 적는다면 에러가 일어날 수 있다.)   
  
**예외(Exception)**  
에러가 일어날 가능성을 개발자가 미리 예상하고 프로그램이 비정상적으로 종료되지 않도록 하는 것. 예외 처리(Exception Handling) 또는 트러블 슈팅(Trouble Shooting)을 통해 예외를 구분하고, 유저에게 문제를 바로잡아달라는 에러 메시지를 띄워줘야 에러로 인한 문제를 예방할 수 있다.  
  
**트래픽(Traffic)**  
서버를 통해 전송되는 데이터의 양. 서비스 방문자 수가 많을수록, 접속이 잦아질수록 증가한다. 트래픽이 너무 높아지면 서버가 먹통이 된다. (Server Down) 흔히 대학교 수강신청 날 학생들이 많이 몰려 수강신청 사이트가 ‘터져’ 버리거나, 외부에서 많은 PC를 감염시켜 의도적으로 특정 서비스가 마비되도록 공격하는 것(DDoS, 분산 서비스 거부 공격) 모두 트래픽 초과와 관련이 깊다. 때문에 백엔드 개발자라면 시스템을 안정적으로 개발해 높은 트래픽에도 서비스 접근이 제한되지 않도록 유지하는 것이 관건.    
  
**데이터베이스(Database, DB)**  
데이터의 집합. 서버를 통해 넘어오는 데이터를 저장하고 관리하는 일종의 창고 역할을 한다. (데이터베이스에 과부하가 걸리면 서버에도 지장을 주고, 관리가 잘못되면 데이터가 날아가는 아찔한 상황이 벌어질 수도 있음)   
  
**데이터베이스 관리 시스템(Database Management System)**  
말 그대로 데이터베이스를 통해 데이터를 저장, 수정, 추출할 수 있도록 관리하는 프로그램. 줄여서 DBMS. (꼭 서버용 프로그램뿐만 데이터베이스인 건 아니다. 개별 PC를 통해 데이터를 저장하는 마이크로소프트의 오피스 툴인 액세스(Access)도 DBMS에 해당한다.)   
  
**관계형 데이터베이스(Relational DataBase)**  
데이터베이스에서 키(key)과 값(value)의 관계(Relationship)로 이루어진 표(table)로 데이터를 구성하는 방식으로, 줄여서 RDB. 오라클(Oracle), MySQL, Microsoft SQL Server 등 가장 널리 쓰이는 데이터베이스 관리 시스템이 관계형 데이터베이스 모델을 따르고 있다.   
  
**쿼리(Query)**  
관계형 데이터베이스에서 데이터를 요청하고 관리하기 위해서는 SQL(Structured Query Language, 구조화 쿼리 언어)이라는 데이터베이스 언어를 써야 한다. SQL을 이용하면 데이터베이스에 쿼리(Query), 즉 질의를 통해 원하는 조건에 맞는 데이터를 추출하거나 조작할 수 있다. 즉 관계형 데이터베이스에서 데이터를 조회, 추출, 조작하기 위해 쓰는 명령어 혹은 질의 자체를 쿼리라고 한다.  
▲ 오라클, Microsoft SQL Server, MySQL은 현재 시장 점유율이 가장 높은 데이터베이스 관리 시스템입니다. 셋 모두 관계형 데이터베이스 모델(RDBMS)이라는 공통점이 있어요.   
  
**API(Application Programming Interface)**  
응용 프로그램 인터페이스. 어떤 응용 프로그램에서 특정한 기능을 사용하기 위해 필요한 데이터를 주고받게끔 만든 도구나 방법을 뜻한다. API가 규격에 맞게 데이터를 요청하고 받아볼 수 있도록 하는 중간 창구 역할을 하는 셈이다. (포털사이트가 기상 관측 기업의 날씨 정보를 받아서 내보내거나, 코로나 확진자 분포를 지도에 띄우거나, 여러 웹사이트에서 페이스북이나 카카오톡 계정을 통한 간편 로그인을 제공하는 것 역시 API를 통해 이루어진다.)   
  
**오픈소스(Open Source)**  
누구나 제한 없이 쓸 수 있는 소스 코드 혹은 소프트웨어. 오픈 소스로 배포된 코드는 열람, 수정, 복제, 재배포 등이 자유롭다. 한국에서는 공공데이터포털(data.go.kr)을 통해 공개된 오픈 API를 내려받을 수 있다.   
▲ 공공데이터포털에서는 공공 API를 누구나 무료로 내려받고, 자유롭게 이용할 수 있어요.   
  
**SDK(Software Development Kit)**  
소프트웨어나 시스템을 만드는 데 쓰이는 개발 도구 키트. SDK 안에는 개발에 필요한 샘플 코드, 코드 편집기 같은 툴이나 콘솔, 안내 문서, API 등이 포함된다. (안드로이드용 모바일 앱을 개발하는 데는 Android SDK가, 자바(Java) 개발을 하는 데는 자바 개발 키트(JDK)가 쓰이는 등 여러 가지 SDK를 찾아볼 수 있다.)   
  
**IDE(Integrated Development Environment)**  
프로그램을 개발하는 데 필요한 소스 코드 작성 및 편집, 컴파일, 디버깅 등 모든 작업을 한번에 할 수 있는 통합 개발 환경. (비주얼 스튜디오(Visual Studio), 이클립스(Eclipse), 인텔리제이 아이디어(IntelliJ Idea) 등.)   
  
**유닉스(Unix)**  
기계어에 가까운 언어인 어셈블리어로 개발한 컴퓨터 운영 체제. 주로 서버 개발이나 시스템 개발에 쓰이며, macOS나 리눅스(Linux), 안드로이드(Android) 등 오늘날 많은 운영 체제의 원형이 되었다.   
  
**리눅스(Linux)**  
유닉스 계열에 속하는 오픈소스 운영체제. 서버 개발 및 구축, 수퍼컴퓨터, 임베디드 기기, 스마트폰 OS 개발 등 다양한 용도로 활발하게 사용되고 있다. 리눅스를 개량해서 만든 배포판도 상당히 많다. (우분투(Ubuntu)와 데비안(Devian)이 대표적.)   
  
**스레드(Thread, 쓰레드)**  
프로그램을 실행하는 하나의 프로세스(Process) 내에서 실제로 작업을 처리하는 주체를 뜻한다. 모든 프로세스는 반드시 한 개 이상의 스레드를 가진다.   
  
**멀티스레드(Multi Thread, 멀티쓰레드)**  
하나의 프로세스가 두 개 이상의 스레드를 가지는 경우. 프로세스 내에 있는 각각의 스레드가 프로세스 메모리를 공유하기 때문에 자원을 절약할 수 있고, 스레드간 정보교환이 쉽다는 장점이 있다. 오늘날 대부분의 최신 운영체제는 멀티스레드 프로그래밍을 지원하며, 시스템 개발, 서버 개발, 응용 프로그램 개발 등 다양한 작업에 쓰이는 기술이다.   
  
**도메인(Domain)**  
숫자로만 이루어진 IP와 달리 inflearn.com처럼 문자로 지정된 인터넷 주소(URL).    
  
**DNS(Domain Name System)**  
도메인을 도메인이 올라가 있는 웹 서버의 IP로 연결하는 시스템. 웹페이지에 접속하기 위해 필요한 IP를 일일이 챙길 필요 없이 도메인과 IP를 변환해주는 역할을 한다.  
▲ 내 IP 주소를 확인하고 싶을 땐 명령 프롬프트(CMD)를 열어 ipconfig를 입력하면 쉽게 알 수 있죠. (윈도 기준) 하지만 내가 접속할 모든 웹 사이트에 맞는 IP를 찾으려고 한다면 벌써부터 한숨이...   
  
**프로토콜(Protocol)**  
컴퓨터나 서버, 통신 장비 등 장치 사이에서 서로 통신하기 위해 미리 정해놓은 규칙. (TCP/IP, HTTPS, HTTP 등이 모두 인터넷 프로토콜이다.)   
  
**HTTP, HTTPS**  
HTTP는 하이퍼텍스트 전송 프로토콜(Hypertext Transfer Protocol)의 약자로 네트워크에서 HTML 문서를 주고받기 위한 통신규약이다. 한편, HTTPS는 HTTP가 가진 보안 취약점을 암호화 및 인증 구조를 통해 개선하기 위해 만들어졌다. (보안 소켓 레이어(SSL, Secure Socket Layer) 개념을 덧붙인 것.)   
  
**동기 방식(Sync.) vs. 비동기 방식(Async.)**  
동기 방식(Synchronous)은 기능에 대한 요청을 보내고 응답을 받아야만 동작이 순차적으로 처리되는 방식이며, 반대로 비동기 방식(Asynchronous)은 요청을 보내는 순서와 상관 없이 동작이 처리되는 방식이다.   
  
**AJAX(Async JavaScript and XML)**  
비동기적 자바스크립트와 XML의 약자로, 한 웹페이지를 로딩할 때 페이지의 일부 데이터만을 갱신하는 웹 개발 기법을 뜻한다. 기존 웹 페이지 로딩 방식은 한 웹사이트 내부에서 페이지를 넘겨 이동할 때마다 다른 웹페이지를 통째로 불러오기 때문에 시간이 오래 걸린다는 단점이 있었다. 하지만 AJAX는 웹페이지의 기본적인 디자인 틀을 갖춰놓은 뒤 자바스크립트를 통해 웹 서버에 필요한 데이터를 요청하고, 디자인 틀 안에서 XML로 이루어진 데이터를 불러와 변경한다. 즉 전체를 로딩할 필요 없이 일부분만 불러오기 때문에 웹페이지를 처리하는 속도가 빨라지고 코드의 양도 줄어든다는 장점이 있다. (비동기 방식) 

**정적 웹 페이지(Static Web Page)**  
서버에 미리 저장된 데이터를 그대로 불러오듯 전달하는 웹 페이지. 서버의 데이터가 바뀌지 않는 한 모든 사용자는 늘 고정된 웹페이지를 받아보게 된다. 주로 서비스 소개페이지처럼 내용이 자주 변경되지 않는 경우 정적 웹 페이지를 이용하는 경우가 많다.   
  
**동적 웹 페이지(Dynamic Web Page)**  
정적 웹 페이지와 반대로, 서버에 저장된 데이터를 그대로 보여주는 것이 아니라 스크립트를 통해 가공을 거쳐 출력하는 웹페이지. 즉 사용자의 인터랙션에 따라 같은 페이지라도 각기 다른 결과를 받아볼 수 있게 된다. (매번 새롭게 정보가 갱신되는 게시판, SNS처럼 결과가 그때그때 바뀐다면? 전부 동적 웹!)   
  
**적응형 웹(Adaptive Web Design)**  
미리 웹 브라우저가 동작할 기기(스마트폰, PC, 태블릿...) 별로 레이아웃을 여러 개 정해놓고 조건에 따라 그 중 하나를 보여주는 웹 디자인 방식.   
  
**반응형 웹(Responsive Web Design)**  
적응형 웹과 달리 하나의 레이아웃이 웹 브라우저에 맞춰지는 형태로, 웹 브라우저의 가로폭이 바뀔 때마다 페이지 내에 있는 콘텐츠의 크기와 배치도 자동으로 맞춰진다.   
  
**인터랙티브 웹(Interactive Web)**  
웹 페이지 내의 콘텐츠가 클릭, 스크롤, 입력 등 사용자의 동작에 따라 상호작용하는(interact) 웹 페이지. 

**MEAN vs. MERN vs. MEVN**  
풀스택 웹 개발을 할 수 있는 대표적인 기술 스택(묶음) 종류. 공통으로 들어가 있는 M, E, N은 각각 데이터베이스를 담당하는 MongoDB(몽고DB), 서버 개발에 필요한 Express.js(익스프레스), 자바스크립트 프로그램을 실행할 수 있게 해주는(런타임; Runtime) Node.js(노드)를 가리킨다. 여기에 프론트엔드를 담당하는 프레임워크 또는 라이브러리만 각각 앵귤러(Angular; MEAN), 리액트(React; MERN), 뷰(Vue.js; MEVN)로 나뉘는 것. 자바스크립트 언어만으로도 풀스택 개발을 할 수 있다는 이유 때문에 풀스택 개발자가 되고 싶은 사람들 사이에서 인기가 많은 기술 스택이다.   
  
▲ 위에서부터 MEAN, MERN, MEVN 스택에 맞게 기술별로 로고를 모아보았어요. 나머지 기술은 동일하지만, 프론트엔드를 관장하는 기술에만 차이가 나죠. (앵귤러, 리액트, 뷰 중 택일!)   
  
**CDN(Content Delivery Network/Content Distribution Network)**  
한국어로 옮기면 콘텐츠 전송 네트워크. 멀리 떨어진 사용자에게 콘텐츠를 더 빠르게 제공할 수 있도록 하는 기술. 원 서버에서 사용자에게 직접 콘텐츠를 전송하는 대신, 사용자와 가까운 서버로 데이터를 분산시켜 제공하는 네트워크 시스템을 말한다. (다양한 콘텐츠를 전달해야 하는 서비스에서는 필수 기술)   
  
**캐시(Cache)**  
데이터베이스, 서버, 세션 관리 등 다양한 장치에서 널리 쓰이는 개념으로, 사용자가 데이터에 빠르게 접근할 수 있도록 데이터를 임시로 저장해둔 장소. 이를 활용하는 것을 캐싱(Caching)이라고 하며, CDN 역시 웹 콘텐츠를 캐싱한 데이터를 사용자에게 전달함으로써 데이터를 더 빠르게, 더 많이 전송하는 기술이다.  
  
**브라우저 캐싱(Browser Caching)**  
웹 페이지에 방문했을 때 페이지를 이루는 이미지나 HTML, CSS 문서 등 일부 데이터를 사용자의 PC에 저장해둔 뒤, 페이지를 재방문했을 때 서버에 요청할 필요 없이 PC에 저장했던 데이터를 불러오는 캐싱 기술.   
  
**세션(Session)**  
웹 브라우저를 통해 서버에 접속해 연결을 끊기 전까지(=브라우저를 종료하기 전까지) 한 브라우저를 통해 사용자가 요청하는 상태 정보를 서버에서 일정하게 유지하는 기술. 한 사이트 내에서 페이지를 이동할 때마다 로그인이 끊기지 않도록 하는 게 대표적인 예다.  
▲ 간혹 웹 서비스를 이용하다 보면 로그인이 풀렸다며 다시 로그인하라는 안내 메시지가 뜰 때가 있죠. 서버가 재시작되거나, 일정 시간이 지나면 세션이 만료되어 자동으로 로그아웃되도록 할 수 있어요.    
  
**쿠키(Cookie)**  
서버에 저장되는 세션과는 달리 사용자의 브라우저에 간단한 정보를 저장하는 작은 데이터 파일로, 키(Key)와 값(Value)의 구조로 이루어진다. 브라우저에 ID나 암호를 저장하거나, 사이트 방문 기록을 저장하는 데 주로 쓰인다. 

**UX(User eXperience)**  
사용자가 서비스를 이용하며 느끼고 얻는 경험.   
  
**UI(User Interface)**  
사용자가 서비스를 이용하는 환경. UI는 사용자 경험(UX)에 큰 영향을 미친다.  
▲ 앱의 UI는 사용자가 어떤 식으로 기능에 접근하는지를 결정하는 역할을 해요. 자연히 사용자 경험(UX)에도 큰 영향을 주는 요소입니다.   
  
**GUI(Graphic User Interface)**  
그래픽 유저 인터페이스. 사용자가 화면 안의 요소를 시각적으로 확인하고, 마우스로 아이콘을 클릭해 기능을 작동시키는 환경. 도스처럼 문자로 된 명령어를 입력시켜야 하는 CLI(Command-Line Interface)에 반대되는 개념이다.   
  
**모달(Modal Window)**  
UI 디자인 요소로, 웹페이지나 프로그램 화면에서 대화상자 형태로 나타나 사용자에게 동작을 요구하는 창. 웹에서는 팝업(Pop-up) 창과 비슷하지만, 브라우저를 새로 띄우는 팝업과 달리 한 브라우저 화면 내부에서 레이어를 한 겹 얹듯이 새로운 창을 보여준다.   
  
**임베디드(Embeded, Embeded System)**  
특정 기능을 수행하는 전자 시스템을 내장해놓은 것. 일반적인 PC가 다양한 목적에 따라 범용적으로 쓰인다면, 임베디드 시스템은 각종 가전제품, ATM, 개표기, 키오스크 등 특수한 목적에 따라 기능을 수행하도록 제작된다. OS는 보통 간소한 형태(예: 임베디드 리눅스)로 탑재되거나 생략되며, 낮은 전력으로도 정교힌 기능을 실시간으로 처리할 수 있게끔 하는 게 핵심이다.   
  
**펌웨어(Firmware)**  
하드웨어를 제어하는 소프트웨어. 하드웨어가 출고될 때부터 포함되어 있는(임베디드) 프로그램인 만큼 하드웨어보다는 교체하기 쉽지만 일반적인 소프트웨어에 비해서는 어렵기 때문에 soft도 hard도 아닌 그 중간쯤의 firm이라는 이름이 붙었다.  
▲ 인터넷을 쓰게 해주는 공유기에도 펌웨어가 깔려 있어요. 치명적인 버그나 보안 문제를 막기 위해 공유기 제조사에서 새로운 버전의 펌웨어를 계속해서 제공하는 만큼, 종종 펌웨어 업그레이드를 해주는 게 좋답니다.    
  
**알고리즘(Algorithm)**  
문제를 해결하기 위해 필요한 계산 절차. 프로그래밍 언어를 통해 알고리즘을 프로그램으로 만들어가는 작업을 프로그래밍이라고 한다.   
  
**자료 구조(Data Structure)**  
데이터를 효과적으로 접근할 수 있도록 만들어진 데이터 체제로, 코드상에서 자료를 저장하는 방법, 자료끼리의 관계 등을 구조적으로 표현하는 방식. 어떤 자료구조를 쓰느냐에 따라 코드 효율이 달라진다. (컴공 학부생의 필수 코스!) 

**디버깅(Debugging)**  
디버깅 또는 디버그(Debug). 프로그래밍 과정에서 발생하는 버그를 찾아 바로잡는 작업. 단순히 버그를 없애는 것뿐만 아니라 문제가 발생한 근본적인 원인을 찾아 해결하는 과정이다.  
  
**리팩토링(Refactoring)**  
이미 완성된 코드의 구조를 개선하는 일. 결과값은 바뀌지 않지만, 코드 내부를 뜯어고쳐 다른 사람이 이해하기 쉽고 간결하게 만드는 행위. 때문에 버그를 고치거나 새로운 기능을 만드는 건 아니지만, 코드의 생산성을 높이고 유지보수가 쉽게끔 만들어준다. (프로그램이 돌아간다고 해서 전부가 아니라는 점!)   
  
**코딩 컨벤션(Coding Convention)**  
지키지 않으면 훗날 다른 사람이 피 보는 것. 여러 개발자가 협업하는 과정에서 서로의 코드를 파악하고 관리하기 쉽도록 일관적인 스타일을 유지하게끔 하는 코드 작성 규칙. 띄어쓰기나 들여쓰기, 함수나 변수의 이름을 지정해주는 방법 등 정해진 코딩 컨벤션을 준수해야 한 프로젝트 안에서 통일된 코드를 짤 수 있다.    
  
**기술 부채(Technical Debt)**  
불확실한 업무, 촉박한 일정에 맞춰 제한된 방법으로 개발을 진행하면서 문제점이 누적된다는 관점. 프로젝트에 대해 정리된 문서를 마련하지 못하거나, 조악한 코드로 작업을 계속하게 되거나, 자동화할 수 있는 일에 대한 자동화 처리를 해놓지 않는 등 다양한 상황에서 기술 부채가 쌓일 수 있다. 마치 이자가 붙듯이 시간이 지나면 지날수록 해결이 까다로워지는 만큼 말 그대로 기술적으로 빚을 지는 셈. 하지만 완전히 피할 수는 없는 일이니만큼 무작정 나쁘다고 보기보다는 관리할 수 있는 선에서 기술부채를 유지하고 작업을 추진해나가는 게 필요하다.  
  
  
**컴파일(Compile), 컴파일러(Compiler), 컴파일드 언어(Compiled Language)**  
사람이 이해할 수 있는 고급 프로그래밍 언어로 작성한 코드를 컴퓨터가 처리할 수 있는 기계어로 한꺼번에 번역하는 과정을 컴파일(Compile)이라고 하는데, 이때 쓰이는 프로그램 또는 시스템이 컴파일러다. 수정이 까다롭지만 프로그램 실행 속도가 빠르다는 장점이 있다. C, C++이 대표적인 컴파일드 언어.   
  
**인터프리트(Interpret), 인터프리터(Interpreter), 인터프리티드 언어(Interpreted Language)**  
인터프리트(interpret)는 컴파일과 달리 프로그램을 실행할 때마다 고급 프로그래밍 언어로 작성된 코드를 한 줄씩 기계어로 번역하는 방식이다. 한 줄 한 줄씩 코드를 실행해내려가기 때문에 실행 속도는 느리지만 디버깅은 쉽다. Javascript, JAVA, Python, PHP 등이 인터프리티드 언어에 속한다.    
  
**깃(Git)**  
깃(Git)은 프로젝트 결과물을 버전별로 백업, 관리하는 툴인 버전 관리 시스템(VCS, Version Control System)의 한 종류로, 프로젝트를 유지보수하면서 발생할 수 있는 문제(실수로 잘못된 코드를 덮어쓴다거나, 다시 활용해야 하는 기존 코드를 날린다거나...) 를 미연에 방지하고 해결하는 역할을 한다.   
  
**깃허브, 깃헙(Github)**  
깃을 기반으로 소스 코드를 온라인에 저장하고, 공유할 수 있게끔 하는 소스코드 관리 서비스. 언제 어디서나 깃허브를 통해 협업 프로젝트를 쉽게 관리할 수 있으며, 뷰(Vue.js), 도커(Docker), 판다스(Pandas) 등 전세계의 다양한 오픈소스 프로젝트가 깃허브를 통해 공개되고 있다. 개발 직군에서 취업하거나 이직할 때 깃허브 계정 링크를 제출하는 경우도 상당수.   
  
▲ 깃허브의 마스코트는 문어 다리에 고양이 얼굴을 한 옥토캣(Octocat)이에요. 깃허브 측에서 트위터의 파랑새 로고를 만든 그래픽 디자이너 사이먼 옥슬리(Simon Oxley)가 그린 일러스트의 독점권을 사들였다고 하네요. ©Github   
  
**레포지토리(Repository)**  
말 그대로 저장소. 파일이나 폴더 등 각종 정보를 저장해두는 공간으로, 깃허브를 통해 개인 또는 팀을 위한 공개/비공개 리포지터리를 무료로 만들 수 있다. 리포, 레포(Repo)로 줄여 부르기도.   
  
**커밋(Commit)**  
깃(Git)에서 코드에 대한 수정사항을 기록하는 명령. 게임을 플레이하다가 세이브 파일을 남기는 것처럼, 커밋을 생성하면 커밋한 시점의 작업 변경 이력을 저장할 수 있다. (몇 년 전부터 개발자들 사이에서는 깃허브 등에 매일매일 자신이 코딩한 내용을 기록하고 공개하는 ‘일일커밋’이 유행하고 있다는 사실!)   
  
**스택오버플로우(Stack Overflow)**  
세계에서 가장 큰 개발자 웹 커뮤니티 중 하나로, 프로그래밍에 대한 다양한 질문/답변이 올라온다. 이름은 프로그래밍 과정에서 데이터가 할당된 메모리 공간을 초과할 때 발생하는 동명의 오류에서 따온 것. 개발자 사이에선 ‘원인을 알 수 없는 버그가 발생하면 스택오버플로를 뒤져보라’는 말까지 돌 정도로 잘 알려진 사이트다.

**오픈소스**

오픈소스란 원래 오픈소스 소프트웨어(Open Source Software, OSS)를 뜻하는 용어입니다. 오픈소스 소프트웨어는 공개적으로 액세스할 수 있게 설계되어 누구나 자유롭게 확인, 수정, 배포할 수 있는 코드입니다. 오픈소스 소프트웨어는 동료 평가(peer review) 와 커뮤니티 기반 프로덕션에 의지하므로, 분산된 동시에 협업 방식으로 개발됩니다. 단일 작성자 또는 기업이 아닌 커뮤니티가 개발하므로 독점적 소프트웨어보다 저렴하고, 유연하며, 지속성이 있습니다.

**디지털 트윈**

디지털 트윈(digital twin)은 미국 [제너럴 일렉트릭](https://ko.wikipedia.org/wiki/%EC%A0%9C%EB%84%88%EB%9F%B4_%EC%9D%BC%EB%A0%89%ED%8A%B8%EB%A6%AD)(GE)이 주창한 개념으로, 컴퓨터에 현실 속 사물의 쌍둥이를 만들고, 현실에서 발생할 수 있는 상황을 컴퓨터로 시뮬레이션함으로써 결과를 미리 예측하는 기술이다.

**트랜젝션**

트랜잭션은 하나의 작업을 수행하기 위해 필요한 데이터베이스의 연산들을 모아 놓은 것으로, 데이터베이스에서 논리적인 작업의 단위이며 장애가 발생했을 때 데이터를 복구하는 작업의 단위이다. 데이터베이스는 정확한 데이터를 유지 및 오류가 발생할 시 빠르게 복구하고, 데이터베이스가 항상 정확하고 일관된 상태를 유지할 수 있도록 다양한 기능을 제공하는데, 그 중심에서 가장 중요한 역할을 하는 것이 트랜잭션이다. 트랜잭션을 관리함으로써 데이터 베이스의 회복과 병행 제어가 가능하기 때문이다.

**아키텍처**

**최적화를 목표로 두고 시스템 구성과 동작원리 그리고 시스템의 구성환경등을 설명 및 설계하는 청사진 또는 설계도** 입니다. 기본 Computer Science 지식을 기반으로 주변환경등을 고려하여, 최상의 소프트웨어를 구성하는 방법을 연구하고 이를 바탕으로 가이드하는 역할을 합니다.

**블록체인의 노드**

노드(node)는 네트워크에서 연결 포인트 혹은 데이터 전송의 종점, 재분배점 등을 의미한다. [블록체인](http://wiki.hash.kr/index.php/%EB%B8%94%EB%A1%9D%EC%B2%B4%EC%9D%B8)은 중앙 집중형 서버에 거래 기록을 보관, 관리하지 않고 거래에 참여하는 개개인의 서버들이 모여 네트워크를 유지 및 관리하는데 이 개개인의 서버, 즉 참여자를 노드라고 한다.

**데브옵스**

데브옵스(DevOps)는 소프트웨어의 개발(Development)과 운영(Operations)의 합성어로서, 소프트웨어 개발자와 정보기술 전문가 간의 소통, 협업 및 통합을 강조하는 개발 환경이나 문화를 말한다. 데브옵스는 소프트웨어 개발조직과 운영조직간의 상호 의존적 대응이며 조직이 소프트웨어 제품과 서비스를 빠른 시간에 개발 및 배포하는 것을 목적으로 한다.

**쿠버네티스&도커**

컨테이너 하나 띄워서 사용해야지 => 도커

0월 0시에 100개의 컨테이너를 자동으로 생성해야지 => 쿠버네티스

도커는 **'기술적인 개념이자 도구'**이고 쿠버네티스는 **'도커를 관리하는 툴'**이라고 생각하면 된다. 이미지를 컨테이너에 띄우고 실행하는 기술이 **도커**고 이런 도커를 기반으로 컨테이너를 관리하는 서비스가 **쿠버네티스**라고 생각하면 된다. 즉, **도커는 '한 개의 컨테이너'**를 관리하는 데 최적, **쿠버네티스는 '여러 개의 컨테이너'를 서비스 단위로 관리하는 데 최적화**되어있다.

도커는 '컨테이너 기반의 오픈소스 가상화 플랫폼'이다. 도커로 컨테이너를 띄운다. 컨테이너란 애플리케이션 & 애플리케이션을 구동하는 환경을 격리한 공간이다. 보통 마이크로 서비스로 사용되며 컨테이너에 프로그램을 띄워서 돌린다**.** 거대한 어플리케이션을 기능별로 나누어 변경/조합이 가능하게 한 컨테이너를 사용하면 하나의 큰 어플을 서비스 단위로 잘라 빠르게 배포가 가능하다. 그리고 각각 분리해서 쓰니 변경사항이 분리된 다른 기능들에 영향 미치지 않는다.

- 코더 : 생각한 바를 코드로 옮길 수 있는 사람

- 프로그래머 : 생각한 바를 공학적인 요소와 함께 사용하여 효율적인 코드로 작성할 수 있는

사람

- 소프트웨어 개발자 : 코딩과 프로그래밍을 베이스로 깔고 실제 프로젝트 진행 시 발생할 수

있는 다양한 이슈를 처리할 수 있는 능력을 갖춘 사람

**아두이노**

**아두이노**([이탈리아어](https://ko.wikipedia.org/wiki/%EC%9D%B4%ED%83%88%EB%A6%AC%EC%95%84%EC%96%B4): Arduino 아르두이노[[\*](https://ko.wikipedia.org/wiki/%EC%9C%84%ED%82%A4%EB%B0%B1%EA%B3%BC:%EC%9D%B4%ED%83%88%EB%A6%AC%EC%95%84%EC%96%B4%EC%9D%98_%ED%95%9C%EA%B8%80_%ED%91%9C%EA%B8%B0)])는 [오픈 소스](https://ko.wikipedia.org/wiki/%EC%98%A4%ED%94%88_%EC%86%8C%EC%8A%A4)를 기반으로 한 단일 보드 [마이크로컨트롤러](https://ko.wikipedia.org/wiki/%EB%A7%88%EC%9D%B4%ED%81%AC%EB%A1%9C%EC%BB%A8%ED%8A%B8%EB%A1%A4%EB%9F%AC)로 완성된 보드(상품)와 관련 개발 도구 및 환경을 말한다. 아두이노는 다수의 스위치나 센서로부터 값을 받아들여, [LED](https://ko.wikipedia.org/wiki/LED)나 모터와 같은 외부 전자 장치들을 통제함으로써 환경과 상호작용이 가능한 물건을 만들어 낼 수 있다. [임베디드 시스템](https://ko.wikipedia.org/wiki/%EC%9E%84%EB%B2%A0%EB%94%94%EB%93%9C_%EC%8B%9C%EC%8A%A4%ED%85%9C) 중의 하나로 쉽게 개발할 수 있는 환경을 이용하여, 장치를 제어할 수 있다. 아두이노 통합 개발 환경([IDE](https://ko.wikipedia.org/wiki/IDE))을 제공하며, 소프트웨어 개발과 실행코드 업로드도 제공한다.[[1]](https://ko.wikipedia.org/wiki/%EC%95%84%EB%91%90%EC%9D%B4%EB%85%B8#cite_note-arduino_IDE-1) 또한 [어도비 플래시](https://ko.wikipedia.org/wiki/%EC%96%B4%EB%8F%84%EB%B9%84_%ED%94%8C%EB%9E%98%EC%8B%9C), [프로세싱](https://ko.wikipedia.org/wiki/%ED%94%84%EB%A1%9C%EC%84%B8%EC%8B%B1_(%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D_%EC%96%B8%EC%96%B4)), Max/MSP와 같은 소프트웨어와 연동할 수 있다.[[2]](https://ko.wikipedia.org/wiki/%EC%95%84%EB%91%90%EC%9D%B4%EB%85%B8#cite_note-Arduino-Intro-2) 오픈소스이기 때문에 아두이노를 기반으로 여러 가지 프로젝트를 수행할 수 있다

**데이터 무결성**

무결성이란 데이터 베이스에 값이 정확하다는 것을 의미한다. 항상 정확한 데이터를 유지하고 있다는 것을 의미한다. 무결성 제약 조건이란 데이터 베이스는 무결성 조건에 따라 항상 정확한 데이터만을 가지고  있어야 한다. 하지만 정확하지 않은 데이터가 입력될 경우 이러한 값이 저장되지 않도록 하는 조건

**노드**(node)

노드는 [컴퓨터 과학](https://ko.wikipedia.org/wiki/%EC%BB%B4%ED%93%A8%ED%84%B0_%EA%B3%BC%ED%95%99)에 쓰이는 기초적인 단위이다. 노드는 대형 네트워크에서는 장치나 데이터 지점(data point)을 의미한다. 개인용 컴퓨터, 휴대전화, 프린터와 같은 정보처리 장치들이 노드이다. 인터넷에서 노드를 정의할 때 노드는 [IP 주소](https://ko.wikipedia.org/wiki/IP_%EC%A3%BC%EC%86%8C)를 보유한 어떠한 것도 될 수 있다. 노드는 더 커다란 [자료 구조](https://ko.wikipedia.org/wiki/%EC%9E%90%EB%A3%8C_%EA%B5%AC%EC%A1%B0)의 일부분 하나 하나를 의미할 수 있는데 이를테면 [연결 리스트](https://ko.wikipedia.org/wiki/%EC%97%B0%EA%B2%B0_%EB%A6%AC%EC%8A%A4%ED%8A%B8)라든지 [트리](https://ko.wikipedia.org/wiki/%ED%8A%B8%EB%A6%AC_%EA%B5%AC%EC%A1%B0) 자료 구조를 들 수 있다. 노드는 [데이터](https://ko.wikipedia.org/wiki/%EB%8D%B0%EC%9D%B4%ED%84%B0)를 포함하며 다른 노드와 연결될 수도 있다. 노드 간 링크는 [포인터](https://ko.wikipedia.org/wiki/%ED%8F%AC%EC%9D%B8%ED%84%B0_(%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D))에 의해 구현되기도 한다.

**레거시 시스템**

**레거시 시스템**(legacy system)은 낡은 기술이나 방법론, [컴퓨터 시스템](https://ko.wikipedia.org/wiki/%EC%BB%B4%ED%93%A8%ED%84%B0_%EC%8B%9C%EC%8A%A4%ED%85%9C), [소프트웨어](https://ko.wikipedia.org/wiki/%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4) 등을 말한다. 이는 현대까지도 남아 쓰이는 기술을 부르는 말일 수도 있지만, 더 이상 쓰이지 않더라도 현대의 기술에 영향을 주는 경우도 포함한다. 많은 소프트웨어 개발자들이 레거시 시스템을 사용하는 것에 잠재적 문제가 있다고 생각한다.[[1]](https://ko.wikipedia.org/wiki/%EB%A0%88%EA%B1%B0%EC%8B%9C_%EC%8B%9C%EC%8A%A4%ED%85%9C#cite_note-1) 예를 들어 오래된 하드웨어를 위해 설계된 소프트웨어의 경우, 새로운 하드웨어에서 실행되도록 하기 위해서 [에뮬레이션](https://ko.wikipedia.org/wiki/%EC%97%90%EB%AE%AC%EB%A0%88%EC%9D%B4%EC%85%98)이나 [하위 호환성](https://ko.wikipedia.org/wiki/%ED%95%98%EC%9C%84_%ED%98%B8%ED%99%98%EC%84%B1)을 구현해줘야 할 수 있다.

**이더리움**

**이더리움**(Ethereum)은 [블록체인](https://ko.wikipedia.org/wiki/%EB%B8%94%EB%A1%9D%EC%B2%B4%EC%9D%B8) 기술을 기반으로 [스마트 계약](https://ko.wikipedia.org/wiki/%EC%8A%A4%EB%A7%88%ED%8A%B8_%EA%B3%84%EC%95%BD) 기능을 구현하기 위한 분산 컴퓨팅 플랫폼이자 플랫폼의 자체 통화명이다. 이더리움이 제공하는 이더(Ether)는 [비트코인](https://ko.wikipedia.org/wiki/%EB%B9%84%ED%8A%B8%EC%BD%94%EC%9D%B8)과 마찬가지로 [암호화폐](https://ko.wikipedia.org/wiki/%EC%95%94%ED%98%B8%ED%99%94%ED%8F%90)의 일종으로 거래되고 있다. 이더리움의 화폐 단위는 ETH로 표시한다. 이더리움은 2015년 7월 30일 [비탈릭 부테린](https://ko.wikipedia.org/wiki/%EB%B9%84%ED%83%88%EB%A6%AD_%EB%B6%80%ED%85%8C%EB%A6%B0)(Vitalik Buterin)이 개발하였다.[[4]](https://ko.wikipedia.org/wiki/%EC%9D%B4%EB%8D%94%EB%A6%AC%EC%9B%80#cite_note-4) 비탈릭 부테린은 가상화폐인 [비트코인](https://ko.wikipedia.org/wiki/%EB%B9%84%ED%8A%B8%EC%BD%94%EC%9D%B8)에 사용된 핵심 기술인 [블록체인](https://ko.wikipedia.org/wiki/%EB%B8%94%EB%A1%9D%EC%B2%B4%EC%9D%B8)에 화폐 거래 기록뿐 아니라 계약서 등의 추가 정보를 기록할 수 있다는 점에 착안하여, 전 세계 수많은 사용자들이 보유하고 있는 컴퓨팅 자원을 활용해 이더리움 [가상 머신](https://ko.wikipedia.org/wiki/%EA%B0%80%EC%83%81_%EB%A8%B8%EC%8B%A0)(EVM)을 만들고, 이 플랫폼을 이용하여 SNS, 이메일, 전자투표 등 다양한 정보를 기록하는 시스템을 창안했다. 이더리움은 C++, 자바, 파이썬, GO 등 주요 프로그래밍 언어를 지원한다.[5]이더리움을 [사물 인터넷](https://ko.wikipedia.org/wiki/%EC%82%AC%EB%AC%BC_%EC%9D%B8%ED%84%B0%EB%84%B7)(IoT)에 적용하면 기계 간 금융 거래도 가능해진다. 예를 들어 고장난 청소로봇이 정비로봇에 돈을 내고 정비를 받고, 청소로봇은 돈을 벌기 위해 정비로봇의 집을 청소하는 것도 가능해진다