개발자들이 자주 쓰는 단어 50개 정리본

BY 이원자탄소

개발자들이 많이 쓰는 단어 50개

초보 개발자를 위한 필수 용어 가이드

들어가는 말

개발을 막 시작했을 때, 용어부터 막히는 경우 정말 많죠. 다른 세상의 언어처럼 느껴질 때도 있고요.
저도 그랬어요. '이게 대체 무슨 말이지?' 하며 검색만 몇 시간을 한 적도 많았거든요.

그래서 준비했어요.
이 자료에는 개발자들이 자주 쓰는 핵심 용어 50가지를 모아서, 최대한 쉽게 풀어 설명했어요.
처음 접하는 사람도 부담 없이 읽을 수 있도록!

목차

1. 변수 (Variable)

변수는 프로그래밍에서 값을 저장하는 데 사용되는 "이름이 지정된 저장 공간"입니다. 예를 들어, `age = 30` 과 같이 사용하면 `age` 라는 변수에 `30` 이라는 값을 저장할 수 있습니다. 변수에 저장된 값은 프로그램이 실행되는 동안 변경될 수 있습니다.

활용 예시: 사용자 이름, 점수, 계산 결과 등을 저장할 때 사용됩니다.

2. 함수 (Function)

함수는 특정 작업을 수행하는 코드 블록에 이름을 붙인 것입니다. 반복되는 작업을 효율적으로 처리하고, 코드를 모듈화하여 재사용성을 높이는 데 사용됩니다. 함수는 입력을 받아(매개변수) 어떤 작업을 수행한 후 결과를 반환(반환 값)할 수 있습니다.

활용 예시: 숫자 계산, 데이터 처리, 특정 UI 요소 생성 등 다양한 기능을 캡슐화할 때 사용됩니다.

3. 클래스 (Class)

클래스는 객체 지향 프로그래밍(OOP)에서 특정 종류의 객체를 생성하기 위한 "청사진" 또는 "템플릿"입니다. 클래스는 객체가 가질 속성(변수)과 행동(메서드)을 정의합니다. 예를 들어, '자동차' 클래스는 '색상', '모델'과 같은 속성과 '가속', '정지'와 같은 행동을 가질 수 있습니다.

활용 예시: 사용자, 제품, 주문 등 실제 세계의 개념을 코드로 표현하고 구조화할 때 사용됩니다.

4. 객체 (Object)

객체는 클래스의 "인스턴스"입니다. 즉, 클래스라는 청사진을 바탕으로 실제로 만들어진 실체입니다. '자동차' 클래스에서 '내 빨간색 스포츠카'라는 특정 객체를 만들 수 있습니다. 객체는 클래스에 정의된 속성들을 실제 값으로 가지고, 메서드를 통해 행동할 수 있습니다.

활용 예시: 특정 사용자의 정보, 특정 제품의 상세 정보 등 개별적인 실체를 다룰 때 사용됩니다.

5. 배열 (Array)

배열은 "같은 종류의 데이터들을 순서대로 모아놓은 자료 구조"입니다. 각 데이터는 고유한 인덱스(위치 번호)를 가지며, 이 인덱스를 통해 특정 데이터에 접근할 수 있습니다. 예를 들어, `[1, 2, 3, 4, 5]` 는 숫자 배열이며, 첫 번째 요소는 인덱스 0에 있습니다.

활용 예시: 학생들의 점수 목록, 쇼핑 카트의 상품 목록 등 여러 개의 관련 데이터를 묶어서 관리할 때 사용됩니다.

6. 문자열 (String)

문자열은 "문자들의 순서 있는 나열"을 의미합니다. 텍스트 데이터를 다룰 때 사용되며, 따옴표(홑따옴표 또는 쌍따옴표)로 묶어서 표현합니다. 예를 들어, `"안녕하세요, 개발자 여러분!"` 은 문자열입니다.

활용 예시: 사용자 이름, 메시지, 파일 경로 등 텍스트 기반의 모든 데이터를 표현할 때 사용됩니다.

7. 정수 (Integer)

정수는 "소수점 이하가 없는 양의 정수, 음의 정수, 그리고 0을 포함하는 숫자"입니다. 프로그래밍에서 가장 기본적인 숫자 데이터 타입 중 하나입니다. 예를 들어, `10`, `-5`, `0` 등이 정수에 해당합니다.

활용 예시: 개수 세기, 나이, 등급, 인덱스 등 정확한 정수 값을 다룰 때 사용됩니다.

8. 부동 소수점 (Float/Double)

부동 소수점은 "소수점을 포함하는 숫자"를 의미합니다. 주로 실수 값을 표현할 때 사용됩니다. `float`은 단정밀도, `double`은 배정밀도로 `double`이 더 정밀한 값을 표현할 수 있습니다. 예를 들어, `3.14`, `-0.5` 등이 부동 소수점에 해당합니다.

활용 예시: 통화 금액, 측정값, 과학 계산 등 소수점이 필요한 정밀한 계산에 사용됩니다.

9. 불리언 (Boolean)

불리언은 "참(true) 또는 거짓(false) 두 가지 값 중 하나"만을 가질 수 있는 데이터 타입입니다. 주로 조건문이나 논리 연산에서 사용되어 프로그램의 흐름을 제어합니다. 예를 들어, `isLoggedIn = true`는 사용자가 로그인했음을 나타냅니다.

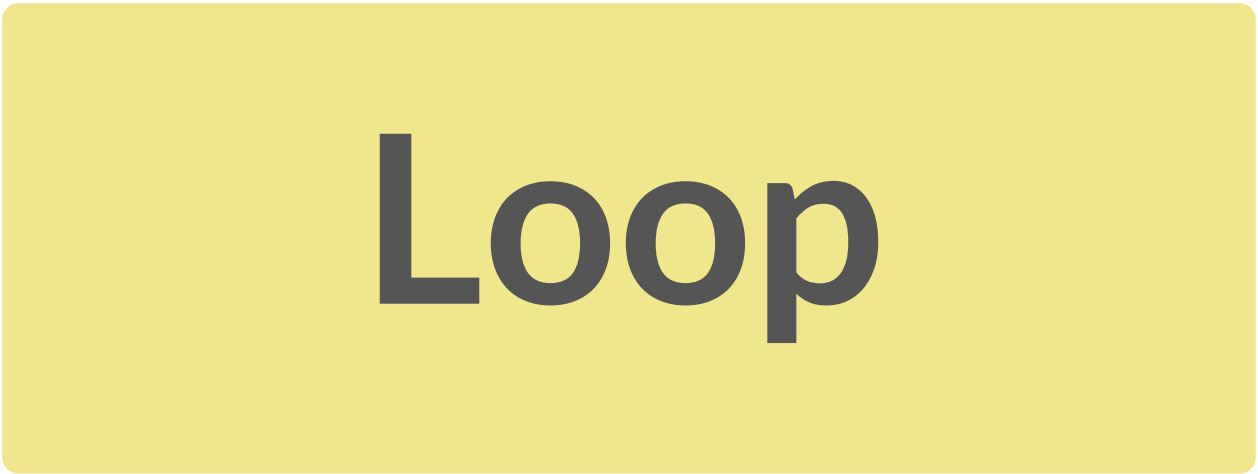
활용 예시: 특정 조건의 충족 여부 확인, 스위치 온/오프 상태, 플래그 설정 등에 사용됩니다.

10. 조건문 (Conditional Statement)

조건문은 "특정 조건이 참일 때만 코드를 실행하도록 제어하는 구문"입니다. 가장 흔한 형태는 `if-else` 문이며, 조건에 따라 다른 동작을 수행하게 합니다. 예를 들어, "만약 날씨가 좋으면 외출하고, 그렇지 않으면 집에서 쉰다."와 같습니다.

활용 예시: 사용자 입력 유효성 검사, 권한 확인, 게임 로직 분기 등 다양한 상황에서 프로그램의 흐름을 결정합니다.

11. 반복문 (Loop)



반복문은 "특정 코드 블록을 여러 번 반복해서 실행하도록 하는 구문"입니다. `for`, `while` 등이 대표적이며, 배열의 모든 요소를 처리하거나 특정 조건이 충족될 때까지 작업을 반복할 때 유용합니다. 예를 들어, 1부터 100까지의 숫자를 출력하는 작업에 사용됩니다.

활용 예시: 리스트의 항목 처리, 데이터베이스의 모든 레코드 접근, 애니메이션 프레임 업데이트 등 반복적인 작업에 사용됩니다.

12. 상수 (Constant)

상수는 "한 번 값이 할당되면 프로그램 실행 중에 변경할 수 없는 값"입니다. 변수와 달리, 상수는 고정된 값을 유지해야 할 때 사용됩니다. 예를 들어, 원주율(π), 지구 중력 가속도와 같이 변하지 않는 값을 정의할 때 사용합니다.

활용 예시: 수학적 상수, 고정된 설정 값, 변경되지 않는 에러 메시지 등을 정의할 때 사용됩니다.

13. 주석 (Comment)

주석은 "코드에 대한 설명이나 메모를 작성하는 부분"으로, 프로그램 실행에는 영향을 미치지 않습니다. 주로 코드의 목적, 복잡한 로직 설명, 임시 비활성화 등을 위해 사용됩니다. 다른 개발자나 미래의 자신을 위해 코드를 이해하는 데 도움을 줍니다.

활용 예시: 함수의 기능 설명, 변수의 용도, 특정 알고리즘의 동작 방식 등을 기록할 때 사용됩니다.

14. 디버깅 (Debugging)

디버깅은 "프로그램 코드에서 오류(버그)를 찾아내고 수정하는 과정"입니다. 디버거라는 도구를 사용하여 코드 실행을 단계별로 추적하고, 변수 값을 확인하며, 예상치 못한 동작의 원인을 분석합니다.

활용 예시: 프로그램이 예상대로 작동하지 않을 때, 에러 메시지가 발생할 때, 논리적 결함을 찾을 때 필수적으로 수행되는 작업입니다.

15. 컴파일 (Compile)

컴파일은 "사람이 작성한 고급 프로그래밍 언어(소스 코드)를 컴퓨터가 이해하고 실행할 수 있는 기계어(목적 코드 또는 실행 파일)로 변환하는 과정"입니다. 컴파일러라는 도구가 이 작업을 수행합니다. 컴파일 과정에서 문법적 오류가 감지되기도 합니다.

활용 예시: C++, Java와 같은 컴파일 언어에서 프로그램 배포 전에 반드시 거쳐야 하는 단계입니다.

16. 인터프리터 (Interpreter)

인터프리터는 "소스 코드를 한 줄씩 읽어들이면서 동시에 실행하는 프로그램"입니다. 컴파일러와 달리 전체 코드를 미리 변환하지 않고, 필요할 때마다 변환하고 실행합니다. 이 때문에 컴파일 과정이 없어 개발 속도가 빠를 수 있지만, 실행 속도는 컴파일된 코드보다 느릴 수 있습니다.

활용 예시: Python, JavaScript, Ruby와 같은 스크립트 언어에서 코드를 실행할 때 사용됩니다.

17. 라이브러리 (Library)

라이브러리는 "자주 사용되는 특정 기능들을 미리 구현해 놓은 코드의 집합"입니다. 개발자가 반복해서 작성해야 하는 코드를 줄여주어 생산성을 높여줍니다. 개발자는 필요한 라이브러리를 가져와서 자신의 프로젝트에 추가하여 사용할 수 있습니다.

활용 예시: 데이터 분석을 위한 NumPy, 웹 개발을 위한 jQuery, 이미지 처리를 위한 Pillow 등이 있습니다.

18. 프레임워크 (Framework)

프레임워크는 "소프트웨어 개발을 위한 기본적인 구조와 규칙을 제공하는 틀"입니다. 라이브러리보다 더 광범위하며, 개발의 전반적인 흐름과 아키텍처를 정의합니다. 프레임워크는 "제어의 역전(Inversion of Control)" 개념을 가지는데, 이는 프레임워크가 개발자의 코드를 호출하고 관리한다는 의미입니다.

활용 예시: 웹 개발을 위한 Spring (Java), Django (Python), React (JavaScript) 등이 대표적인 예시입니다.

19. API (Application Programming Interface)

API는 "두 소프트웨어 구성 요소가 서로 통신할 수 있도록 하는 규칙 집합"입니다. 예를 들어, 날씨 앱이 일기 예보를 보여주기 위해 날씨 서비스의 API를 호출하는 식입니다. API는 함수, 프로토콜, 데이터 형식 등을 포함할 수 있습니다.

활용 예시: 지도 서비스 연동, 소셜 로그인 기능 구현, 결제 시스템 통합 등 외부 서비스와의 상호작용에 사용됩니다.

20. 데이터베이스 (Database)

데이터베이스는 "체계적으로 조직화된 데이터의 집합"입니다. 데이터를 효율적으로 저장, 관리, 검색하기 위해 사용됩니다. 관계형 데이터베이스(RDBMS)와 NoSQL 데이터베이스 등 다양한 종류가 있습니다.

활용 예시: 웹사이트 사용자 정보, 쇼핑몰 상품 정보, 은행 거래 내역 등 대량의 구조화된 데이터를 저장하고 관리할 때 사용됩니다.

21. 쿼리 (Query)

쿼리는 "데이터베이스에 정보를 요청하거나 조작하기 위해 사용되는 명령어"입니다. 가장 흔한 쿼리 언어는 SQL(Structured Query Language)입니다. `SELECT`, `INSERT`, `UPDATE`, `DELETE` 등이 주요 SQL 쿼리 명령어입니다.

활용 예시: 특정 조건에 맞는 사용자 목록 가져오기, 새로운 상품 정보 추가, 기존 데이터 수정 등에 사용됩니다.

22. 스키마 (Schema)

데이터베이스 스키마는 "데이터베이스의 구조와 조직을 정의하는 청사진"입니다. 테이블의 이름, 각 테이블의 열(컬럼) 이름, 데이터 타입, 관계, 제약 조건 등을 포함합니다. 데이터가 어떻게 저장되고 서로 연결되는지를 나타냅니다.

활용 예시: 데이터베이스를 설계할 때, 어떤 정보를 어떤 형식으로 저장할지 미리 정의하는 데 사용됩니다.

23. 테이블 (Table)

관계형 데이터베이스에서 테이블은 "행(Row)과 열(Column)로 구성된 데이터를 저장하는 기본 단위"입니다. 각 테이블은 특정 주제(예: 사용자, 제품)에 대한 데이터를 담으며, 여러 테이블이 서로 관계를 맺어 전체 데이터베이스를 구성합니다.

활용 예시: '사용자' 테이블에는 사용자 ID, 이름, 이메일 등의 정보가, '제품' 테이블에는 제품 ID, 이름, 가격 등의 정보가 저장됩니다.

24. 필드 (Field)

데이터베이스 테이블에서 필드(또는 컬럼, 열)는 "특정 종류의 데이터를 저장하는 단위"입니다. 각 필드는 고유한 이름과 데이터 타입(예: 문자열, 정수, 날짜)을 가집니다. 예를 들어, '사용자' 테이블의 '이름', '이메일', '생년월일' 등이 필드에 해당합니다.

활용 예시: 사용자 정보 입력 시, 각 입력란이 테이블의 특정 필드에 매핑됩니다.

25. 레코드 (Record)

데이터베이스 테이블에서 레코드(또는 행, Row)는 "단일 엔티티에 대한 완전한 정보"입니다. 테이블의 각 행은 특정 인스턴스(예: 한 명의 사용자, 하나의 제품)에 대한 모든 필드 값을 포함합니다. 예를 들어, '사용자' 테이블의 한 레코드는 한 사용자의 ID, 이름, 이메일 등 모든 정보를 담고 있습니다.

활용 예시: 특정 사용자의 모든 정보를 한 번에 조회하거나, 새로운 사용자 정보를 데이터베이스에 추가할 때 사용됩니다.

26. 인증 (Authentication)

인증은 "어떤 주체의 신원을 확인하는 과정"입니다. 주로 사용자가 자신이 주장하는 사용자인지 확인하는 절차를 의미합니다. 비밀번호, 지문, 생체 인식 등을 통해 이루어집니다. 예를 들어, 웹사이트에 로그인할 때 아이디와 비밀번호를 입력하는 과정이 인증입니다.

활용 예시: 웹사이트 로그인, 모바일 앱 로그인, API 요청 시 사용자 신원 확인 등에 사용됩니다.

27. 인가 (Authorization)

인가는 "인증된 사용자가 특정 자원이나 기능에 접근할 권한이 있는지 확인하는 과정"입니다. 즉, '누구인지' 확인하는 것이 인증이라면, '무엇을 할 수 있는지' 확인하는 것이 인가입니다. 예를 들어, 일반 사용자는 게시물을 읽을 수 있지만, 관리자만 게시물을 삭제할 수 있는 것이 인가에 해당합니다.

활용 예시: 사용자 역할 기반 접근 제어(RBAC), 특정 데이터에 대한 읽기/쓰기 권한 설정 등에 사용됩니다.

28. 해싱 (Hashing)

해싱은 "임의의 길이를 가진 데이터를 고정된 길이의 데이터(해시 값 또는 해시 코드)로 변환하는 과정"입니다. 주로 데이터의 무결성을 검증하거나 비밀번호를 안전하게 저장하는 데 사용됩니다. 해시 함수는 단방향으로, 해시 값으로부터 원본 데이터를 역산하기 어렵도록 설계됩니다.

활용 예시: 비밀번호 저장(원본 비밀번호 대신 해시 값 저장), 파일 다운로드 시 무결성 검증, 데이터 구조(해시 테이블)에서 빠른 데이터 검색 등에 사용됩니다.

29. 암호화 (Encryption)

암호화는 "데이터를 특정 알고리즘(암호화 키 사용)을 통해 읽을 수 없는 형태로 변환하는 과정"입니다. 이는 데이터를 안전하게 전송하거나 저장하여 비인가된 접근으로부터 보호하는 데 사용됩니다. 암호화된 데이터는 복호화 과정을 거쳐 다시 원본 데이터로 복원될 수 있습니다.

활용 예시: 민감한 정보(예: 신용카드 번호) 전송, 통신 보안(HTTPS), 개인 정보 저장 등에 사용됩니다.

30. 복호화 (Decryption)

복호화는 "암호화된 데이터를 원래의 읽을 수 있는 형태로 되돌리는 과정"입니다. 암호화에 사용된 키(또는 관련 키)를 사용하여 데이터를 해독합니다. 암호화와 복호화는 데이터 보안의 핵심 요소입니다.

활용 예시: 암호화된 메시지를 수신했을 때, 이를 읽기 위해 복호화 과정을 거칩니다.

31. 서버 (Server)

서버는 "클라이언트에게 서비스나 자원을 제공하는 컴퓨터 프로그램 또는 장치"입니다. 웹 서버, 데이터베이스 서버, 파일 서버 등 다양한 종류가 있으며, 클라이언트의 요청을 처리하고 응답을 보냅니다. 예를 들어, 웹사이트에 접속할 때 웹 페이지를 제공하는 것이 웹 서버입니다.

활용 예시: 웹 페이지 호스팅, 데이터베이스 관리, 게임 서버 운영, 파일 저장 및 공유 등에 사용됩니다.

32. 클라이언트 (Client)

클라이언트는 "서버에 서비스나 자원을 요청하고 사용하는 컴퓨터 프로그램 또는 장치"입니다. 웹 브라우저, 모바일 앱, 데스크톱 애플리케이션 등이 클라이언트의 예시입니다. 클라이언트는 사용자의 상호작용을 처리하고 서버로부터 받은 정보를 사용자에게 표시합니다.

활용 예시: 웹 브라우저로 웹사이트에 접속하거나, 스마트폰 앱으로 데이터를 요청하는 모든 행위의 주체입니다.

33. 프로토콜 (Protocol)

프로토콜은 "네트워크를 통해 데이터를 주고받을 때 컴퓨터들이 서로 통신하기 위해 사용하는 규칙과 절차의 집합"입니다. 데이터의 형식, 전송 방식, 에러 처리 방법 등을 정의합니다. HTTP, FTP, TCP/IP 등이 대표적인 네트워크 프로토콜입니다.

활용 예시: 웹 브라우저와 서버 간의 통신(HTTP), 파일 전송(FTP), 이메일 송수신(SMTP, POP3, IMAP) 등에 사용됩니다.

34. HTTP (Hypertext Transfer Protocol)

HTTP는 "월드 와이드 웹에서 데이터를 주고받는 데 사용되는 핵심 프로토콜"입니다. 웹 브라우저가 웹 서버로부터 웹 페이지나 다른 웹 자원을 가져올 때 사용됩니다. HTTP는 무상태(stateless) 프로토콜로, 각 요청이 독립적으로 처리됩니다.

활용 예시: 웹사이트 접속 시 URL 앞에 붙는 `http://` 가 이 프로토콜을 사용한다는 의미입니다.

35. HTTPS (Hypertext Transfer Protocol Secure)

HTTPS는 "HTTP에 보안 기능(SSL/TLS 암호화)이 추가된 프로토콜"입니다. 클라이언트와 서버 간에 전송되는 데이터를 암호화하여 중간에서 가로채거나 변조하는 것을 방지합니다. 개인 정보나 금융 정보와 같은 민감한 데이터를 다루는 웹사이트에서 필수로 사용됩니다.

활용 예시: 온라인 뱅킹, 쇼핑몰 결제, 로그인 페이지 등 보안이 중요한 모든 웹 서비스에서 사용됩니다.

36. URL (Uniform Resource Locator)

URL은 "인터넷상의 특정 자원(웹 페이지, 이미지, 파일 등)의 위치를 나타내는 주소"입니다. 웹 브라우저에서 웹사이트에 접속할 때 주소창에 입력하는 것이 URL입니다. 프로토콜, 도메인 이름, 포트 번호, 경로, 쿼리 매개변수, 프래그먼트 등으로 구성됩니다.

활용 예시: `https://www.google.com/search?q=개발자` 와 같이 웹 리소스에 접근할 때 사용됩니다.

37. URI (Uniform Resource Identifier)

URI는 "인터넷상의 자원을 식별하는 문자열"을 의미합니다. URL은 URI의 하위 개념으로, 자원의 위치를 지정하는 반면, URI는 자원을 식별만 할 수도 있습니다(위치를 지정하지 않고). 모든 URL은 URI이지만, 모든 URI가 URL인 것은 아닙니다.

활용 예시: 웹 자원을 식별하는 데 사용되며, URL을 포함한 더 넓은 개념입니다.

38. 리포지토리 (Repository)

리포지토리는 "버전 관리 시스템(VCS)에서 프로젝트의 모든 파일과 변경 이력을 저장하는 중앙 저장소"입니다. Git과 같은 버전 관리 시스템에서 코드를 관리하고 협업하는 데 사용됩니다. 로컬 리포지토리(내 컴퓨터)와 원격 리포지토리(GitHub, GitLab 등)로 나뉩니다.

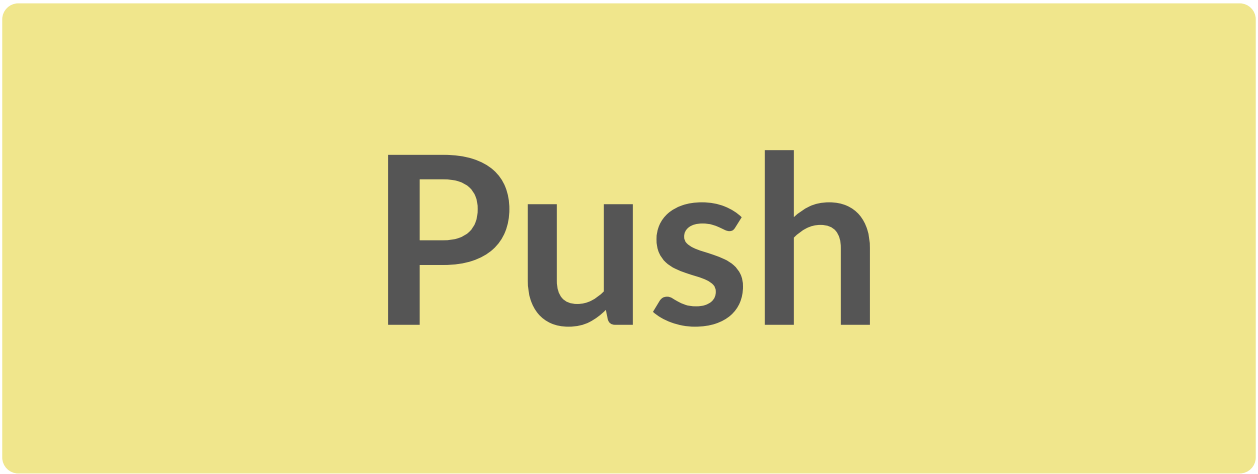
활용 예시: 프로젝트의 소스 코드, 문서, 이미지 등 모든 자산을 관리하고 버전별로 추적하는 데 사용됩니다.

39. 커밋 (Commit)

커밋은 "버전 관리 시스템에서 코드 변경 사항을 저장소에 기록하는 행위"입니다. 각 커밋은 특정 변경 사항의 스냅샷을 나타내며, 고유한 ID와 변경 내용에 대한 설명(커밋 메시지)을 가집니다. 커밋을 통해 언제 누가 무엇을 변경했는지 추적할 수 있습니다.

활용 예시: 기능 구현 완료, 버그 수정, 문서 업데이트 등 코드 변경 사항을 저장할 때 사용됩니다.

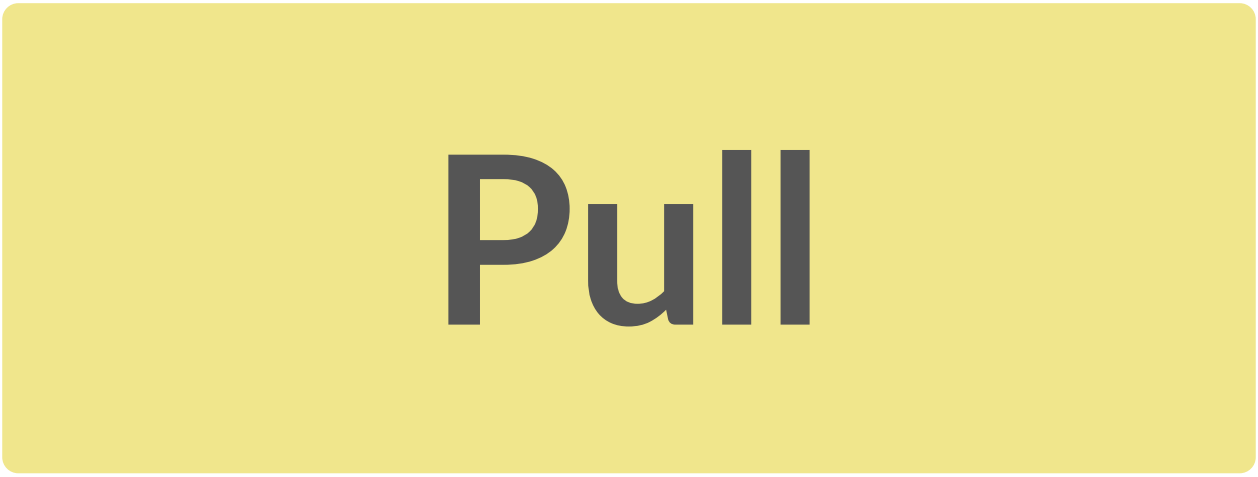
40. 푸시 (Push)



푸시는 "로컬 리포지토리의 커밋된 변경 사항을 원격 리포지토리로 업로드하는 행위"입니다. 이를 통해 다른 팀원들과 코드 변경 사항을 공유할 수 있습니다. `git push` 명령어를 사용하여 수행됩니다.

활용 예시: 로컬에서 작업한 코드를 팀원들이 볼 수 있도록 중앙 저장소에 반영할 때 사용됩니다.

41. 풀 (Pull)



풀은 "원격 리포지토리의 최신 변경 사항을 로컬 리포지토리로 다운로드하고 병합하는 행위"입니다. 다른 팀원이 푸시한 변경 사항을 내 로컬 환경에 반영하여 최신 상태를 유지할 때 사용됩니다. `git pull` 명령어를 사용하여 수행됩니다.

활용 예시: 팀원들의 작업 내용을 내 컴퓨터로 가져와서 자신의 작업과 합칠 때 사용됩니다.

42. 브랜치 (Branch)

브랜치는 "코드 베이스의 독립적인 개발 라인"입니다. 기존 코드에 영향을 주지 않으면서 새로운 기능 개발, 버그 수정 등을 할 때 사용됩니다. 작업이 완료되면 메인 브랜치로 병합(merge)할 수 있습니다.

활용 예시: 새로운 기능 개발 시 `feature/login` 브랜치를 생성하여 작업하고, 버그 수정 시 `bugfix/issue-123` 브랜치를 생성하여 작업합니다.

43. 병합 (Merge)

병합은 "두 개 이상의 브랜치에서 변경된 내용을 하나로 합치는 과정"입니다. 예를 들어, 개발이 완료된 기능 브랜치의 내용을 메인 브랜치로 통합할 때 사용됩니다. 병합 과정에서 충돌이 발생할 수 있습니다.

활용 예시: 여러 개발자가 각자 작업한 내용을 최종적으로 하나의 코드 베이스로 합칠 때 사용됩니다.

44. 충돌 (Conflict)

충돌은 "두 브랜치에서 같은 파일의 같은 부분을 다르게 변경했을 때, 버전 관리 시스템이 자동으로 병합할 수 없어 수동 개입이 필요한 상황"입니다. 개발자는 충돌된 부분을 직접 확인하고 올바른 내용으로 수정해야 합니다.

활용 예시: 여러 명이 같은 파일의 동일한 줄을 수정했을 때 주로 발생하며, 이를 해결하는 것이 개발자의 중요한 업무 중 하나입니다.

45. 배포 (Deployment)

배포는 "개발된 소프트웨어를 사용자들이 접근하고 사용할 수 있도록 서버나 사용자 기기에 설치하고 실행 환경을 구성하는 과정"입니다. 웹 애플리케이션의 경우, 코드를 서버에 올리고 웹 서버를 설정하는 등의 작업을 포함합니다. CI/CD(지속적 통합/지속적 배포) 파이프라인의 핵심 단계입니다.

활용 예시: 새로운 웹사이트를 온라인에 공개하거나, 업데이트된 앱 버전을 앱 스토어에 출시할 때 배포 과정이 필요합니다.

46. 로깅 (Logging)

로깅은 "소프트웨어의 실행 중 발생하는 이벤트나 상태 정보를 기록하는 과정"입니다. 주로 디버깅, 시스템 모니터링, 오류 분석, 사용자 행동 분석 등에 활용됩니다. 로그는 파일, 데이터베이스, 콘솔 등 다양한 형태로 저장될 수 있습니다.

활용 예시: 사용자 로그인 실패, 데이터베이스 연결 오류, 특정 함수의 실행 시간 등을 기록하여 문제 해결 및 성능 개선에 사용됩니다.

47. 테스트 (Testing)

테스트는 "소프트웨어가 요구 사항에 따라 올바르게 작동하는지 확인하고, 오류나 결함을 찾아내는 과정"입니다. 다양한 종류의 테스트(단위 테스트, 통합 테스트, 시스템 테스트 등)가 있으며, 소프트웨어의 품질과 안정성을 보장하는 데 필수적입니다.

활용 예시: 개발자가 코드를 작성한 후 해당 코드가 예상대로 작동하는지 검증하거나, QA(품질 보증) 팀이 전체 시스템의 기능을 확인하는 데 사용됩니다.

48. 유닛 테스트 (Unit Test)

유닛 테스트는 "소프트웨어의 가장 작은 단위(함수, 메서드, 클래스 등)가 올바르게 작동하는지 개별적으로 검증하는 테스트"입니다. 일반적으로 개발자가 직접 작성하며, 코드 변경 시 빠르게 회귀 오류를 감지하는 데 도움을 줍니다.

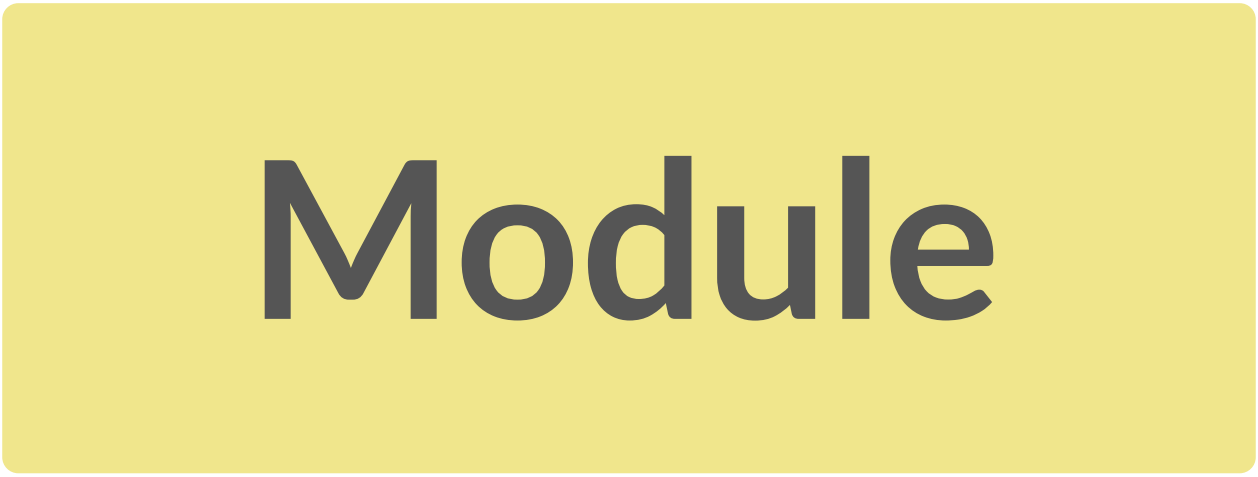
활용 예시: 특정 계산 함수가 올바른 결과를 반환하는지, 특정 유틸리티 메서드가 예상대로 동작하는지 등을 확인하는 데 사용됩니다.

49. 통합 테스트 (Integration Test)

통합 테스트는 "개별적으로 테스트된 여러 모듈이나 구성 요소들이 함께 작동할 때 올바르게 상호작용하는지 검증하는 테스트"입니다. 예를 들어, 사용자 인증 모듈과 데이터베이스 연동 모듈이 함께 작동할 때 문제가 없는지 확인하는 것입니다.

활용 예시: 여러 API 엔드포인트가 서로 연동되는 방식, 프론트엔드와 백엔드 간의 데이터 흐름 등을 테스트할 때 사용됩니다.

50. 모듈 (Module)



모듈은 "소프트웨어의 특정 기능이나 로직을 캡슐화한 독립적인 코드 단위"입니다. 코드를 재사용하고, 가독성을 높이며, 유지보수를 용이하게 하기 위해 사용됩니다. 각 모듈은 고유한 목적을 가지며, 다른 모듈과의 의존성을 최소화하는 것이 좋습니다.

활용 예시: 사용자 인증 모듈, 결제 처리 모듈, 데이터베이스 접근 모듈 등 특정 기능을 담당하는 코드 덩어리를 의미합니다.