1.1 핵심개념 이해하기

* 노드 공식설명

Node.jsⓇ는 Chrome V8 Javascript 엔진으로 빌드된 Javascript 런타임입니다.

* 서버로만 활용되는 것이 아님.
* 서버외 런타임으로 사용하는 방법도 배운다.
* 서버 : 네트워크를 통해 클라이언트에 정보나 서비스를 제공하는 컴퓨터 또는 프로그램
* 노드는 자바스크립트 런타임이다.
  + 런타임 : 특정 언어로 만든 프로그램들을 실행할 수 있는 환경
    - 즉, 노드는 자바스크립트의 실행기다.
  + 원래 자바스크립트는 브라우저에서만 동작하였고, 속도문제로 다른 환경에선 돌리는 게 크게 좋지 않았음
    - -> 근데 오픈소스인 구글의 크롬 V8 엔진의 출시로 빠르게 JS 실행이 가능해졌고, 노드 프로젝트가 시작됨
  + 노드는 V8 과 libuv라는 라이브러리 사용. 이 V8과 libuv 는 c, c++로 구현되어있음
  + libuv라이브러리는 노드의 특성인 이벤트기반, 논 블로킹 I/O 모델을 구현하고 있음
* 이벤트 기반(event-driven)
  + 이벤트가 발생할 때 미리 지정해둔 작업을 수행하는 방식
  + 이를 이벤트리스너에 콜백함수를 등록하는 거라고 표현한다.
  + anonymous 함수 : 처음 실행 시의 전역 컨텍스트를 의미
    - 컨텍스트는 함수가 호출되었을 때 생성되는 환경??
    - 호출스택에서 이 함수까지 실행을 완료하면 스택이 비어짐
  + 이벤트 루프 : 이벤트 발생시 호출할 콜백함수관리, 호출된 콜백함수의 실행순서 결정
    - 노드가 종료될 때까지 이벤트 처리를 위한 작업을 반복하므로 loop라고 함
  + 백그라운드 : 타이머나 이벤트리스너들이 대기하는 곳. 여러작업이 동시에 실행될 수 있음
  + 태스크 큐 : 이벤트 발생후 백그라운드에서 여기로 타이머나 이벤트리스너들의 콜백 함수를 보냄.
    - 정해진 순서대로 콜백들이 줄을 서있으므로 콜백큐라고도 불림
  + 이벤트 루프는 호출 스택이 비어있으면 태스크 큐에서 함수를 하나씩 가져와 호출스택에 넣고 실행함 -> 즉 호출스택이 다 끝나야 이벤트로 발생한 함수가 실행됨.
    - **이것이 setTimeout 의 시간이 아주 정확하지 않은** 이유
* 논 블로킹 I/O
  + 자바스크립트 코드는 동시에 실행될 수 없으나, I/O 작업은 동시에 처리될 수 있음
  + 논블로킹 : 이전 작업이 완료될 때가지 대기하지 않고, 다음작업을 수행함
  + 노드는 I/O 작업을 백그라운드로 넘겨 동시에 처리함
  + 하지만, 동시에 처리될 수 잇는 작업을 블로킹방식으로 코딩하면 시간이 오래걸림
    - 논블로킹으로 처리하기 예시
      * setTimeout(longRunningTask, 0); : 이렇게하면 논블로킹으로 실행됨.
    - 하지만 우리가 작성항 코드는 ‘동시에’ 실행할 수 없음.
      * 위 코드는 단지, 오래걸리는 작업 때문에 간단한 작업들이 막히는 상황을 해소한 것뿐.
      * 동시성과 논블로킹은 다르다!
      * 동시성은 동시처리가 가능한 작업을 논블로킹처리해야 얻을 수 있음.
* 싱글스레드
  + 노드는 싱글스레드를 사용하여, 우리가 작성한 코드는 동시에 실행될 수 없음
  + 노드를 실행하면 프로세스가 생성되고 -> 이 프로세스는 여러 스레드를 실행
    - -> 하지만 우리가 사용할 수 있는 스레드는 단 하나.
    - -> 따라서 노드는 싱글스레드라고 여겨진다.
  + 따라서 블로킹이 심하게 일어나는 작업을 처리하지만 않으면 됨 -> 논블로킹으로 처리
  + 노드가 싱글스레드로 동작하지 않는 두 경우 : 스레드풀, 워커스레드
    - 스레드풀 : 특정한 작업에서 노드가 스스로 멀티스레드를 수행하는 것.(파일 입출력 등)
    - 워커스레드 : 노드12버전에서는 우리가 직접 멀티스레드 처리 가능. CPU연산이 많을 때 사용
  + 싱글스레드 블로킹 : 한명한명 전부 완료를 시켜야하기에 비효울적
  + 싱글스레드 논블로킹 : 간단한 작업은 미리 다 처리하고, 딴데서 처리한 결과를 다시 돌려줌
  + 멀티스레드 블로킹 : 한명을 한 스레드가 맡음. 하지만 비용이 많이 발생
  + 멀티스레드 논블로킹 : 가능하지만, 코딩이 어려움. 따라서 노드에서는 멀티프로세스 논블로킹 채택
    - 멀티 프로세싱은 I/O 작업이 많을 때 유용하다.

1.2 서버로서의 노드

* 장단점 : 싱글스레드 논블로킹 모델의 장단점과 비슷하다
  + 노드는 i/o 요청은 libuv 라이브러리에서 논블로킹으로 처리하기에 버틸 수 있음
  + 하지만 CPU부하가 심한 작업은 힘듬.
  + 하나뿐인 스레드가 죽어버리면 서버 전체가 멈춤
* 따라서 노드는 개수는 많이만, 크기는 작은 데이터를 실시간으로 주고받는데 적합하다.
  + 실시간 채팅 어플리케이션, 주식차트 등등
* 노드12에서 멀티스레드가 가능은 하지만, 다른 언어에 비해 속도는 많이 느리다.
* 극단적으로 성능이 필요한 곳이 아니면, 생산성문제로 노드가 좋다.

1.3 서버 외의 노드

* 자바스크립트 런타임이므로 다른 곳에서도 쓸 수 있음
* 웹, 모바일, 어플개발에도 사용됨
  + ex) 리액트, 앵귤러, 리액트 네이티브, 일렉트론(데스크톱 개발도구, atom, slack 등)

2장