Node.js 란?

자바스크립트로 서버를 개발할 수 있다고 하는데 과연 어떻게 되는 것일까요?

웹의 역사부터 알아보면 1990년 Tim Berners lee가 WEB을 창시했습니다.

처음에 웹은 멈춰져있는 정적인 체계의 웹이였습니다. Marc Andreessen 에 의해 Netscape 라는 대중적 웹브라우저가 등장하게 되었고 Brendan Eich에 의해 JavaScript 등장하고 웹에 동적인 체계를 탑재하게 되었습니다. 즉, 사용자와 상호작용이 가능해졌다는 이야기입니다.

WEB이라는 울타리안에 갇혀있던 JavaScript, 대중성을 중시해서 천대 받던 JavaScript가 재조명 받게되는 계기가 있었다. 2004 년, Gmail 등장이다. 순수한 웹기술(HTML, JavaScript 등)을 통해서 만들었음에도 불구하고 뛰어난 성능을 보인 것이다. 이어서 GMap도 순수 웹기술로 구축되었고 이 또한 개발자들에게 신선한 충격을 주었다. 이 후로 자바스크립트의 성장세는 계속된다.

2008년 구글이 Chrome의 성능 향상을 위해 JavaScript Engine 개발하였다. 그것이 V8 이고 이것을 오픈소스로 공개하여 수많은 개발자들을 이끌었다. 그리고 2009년 Ryan Dal 이 자바스크립트 언어로 구현된 서버 사이드 언어 Node.js 를 내보인다.

그렇다면, Web Browser 에서 작동하는 JavaScript와 Node.js 차이는 무엇인가

JavaScript라는 단어에는 두가지의 의미가 혼재되어있다. language 로서의자바스크립트와 Run Time(언어가 작동하는 환경)으로의 자바스크립트이다. 프로그래밍 세계에서 잠깐 벗어나서 실생활의 구체적인 예로 접근을 해보자. 자바스크립트 언어를 '한국어(일상 우리가 사용하는 언어)'라고 한다면 그 한국어를 통해서 병원엘 가거나 법원에 가는 등에 가서 일을 처리할 수 있다. 즉, 자바스크립트란 프로그래밍 언어로 Web Browser를 제어하거나 서버를 제어할 수 있다. 둘 다 JavaScript의 문법을 기반으로 한다. 하지만 다른 함수를 사용한다. 예를 들면 alert 라는 함수는 Only Web에서 작동하는 함수이며 서버에서는 사용할 수 없다. alert이란 함수는 Node.js 라는 런타임에는 없는 함수라는 말이 된다. Node.js 에 alert이라는 명령어를 입력하는 것은 법원에 가서 아프니까 약달라고 하는 꼴이다.

그러므로 자바스크립트라는 언어만 안다고 두 가지를 모두 제어할 수 있는 것이 아니라 각각이 어떠한 기능을 가지고 있는지 알고 있어야 제어가 가능하다. 병원이 뭘 하는 곳인지, 어떤 기능을 갖고 있는지 알아야 하는 것처럼 말이다. Node.js 공부의 시작은 어떠한 기능들이 있는지 알아가는 것이다.

Web과 Node.js는 서로 협력적인 관계이다.

이 두 가지를 제어해서 하나의 완결된 웹 애플리케이션을 만들 수 있다.

Node와 비슷한 언어는 어떤 것이 있을까, 경쟁자라고도 할 수 있겠다.

= 파이썬 루비 PHP Java ... ?

그렇다면 이 경쟁자들과 비교해서 **Node.js의 장점**은?

이 많은 언어 중에 Node.js 를 선택한 개발자가 얻는 **장점**은 무엇이란 말인가?

1. V8 Engine 이다.

구글이 망하지 않는 한 이 엔진은 끈임없이 개선되고 발전할 것이며

지금도 충분한 성능을 보이고 있지만

추후 더 훌륭한 퍼포먼스를 보일 것으로 예상된다.

2. Event - driven 방식

사용자가 이벤트를 발생시켰을 때,

즉, 입력장치로 데이터를 전송했을 때'에만' 작동하는 방식이다.

발생한 이벤트에 대해서만 웹서버가 '연결'을 해주기 때문에 자원을 최소화할 수 있다.

대부분의 웹 서버는 사용자가 이벤트를 발생하기까지를 기다리면서

'자원'(대기시간 / 메모리)을 계속 소비하게 된다.

3. non - blocking 패러다임

non - Blocking I/O1 을 이해하기 위해서 우선 Blocking I/O 방식을 이해하자.

Blockign I/O 방식(동기식 I/O)은 Read/Write 이벤트가 발생하면 이벤트가 끝날때까지 해당 모듈을 점유하게 된다. 즉 다른일을 못하게 된다.

또한 메모리 버퍼에 데이터를 차지하게 되므로 메모리도 소비하게 된다.

요청한 I/O(DB,File,Network)가 완료될 때까지 해당 Thread를 '대기 모드'로 전환 시켰다가 요청한 I/O 완료 후 유저 코드를 실행시킨다.

Blocking 방식의 비효율성을 극복하고자 만들어진 것이 Non-Blocking 방식이다.

I/O작업을 진행하는 동안 유저 프로세스의 작업을 중단시키지 않는다.

non-Blocking I/O(비동기식 I/O)의 경우 Read/Write 이벤트가 시작하자마자 모듈을 변환시켜 다른 작업을 하도록 준비상태가 된다. 그래서 속도가 동기식보다 빠르고 메모리도 덜 차지하게 된다.

적합한 경우에는 굉장히 빠른 퍼포먼스를 보이고

적재적소에 이 방식을 사용하면 퍼포먼스가 크게 향상할 수 있다.

4. Single Thread

양날의 검이다. 장점이라기보다는 특징에 가깝다.

적은 양의 자원으로 일을 처리하는 것이 가능하다는 장점이다.

하지만 만일 어느 한곳에 예외상황 및 에러가 발생한다면 애플리케이션 전체에 영향이 가게 된다.

이렇게 여러 가지의 장점들을 가진 Node.js 이다.

하지만 Node.js 가 갖는 가장 강력한 장점은 **클라이언트와 서버에서의 언어가 동일**하다는 점일 것이다.

조직원들이 같은 언어를 사용하는 것은 굉장한 매력이라고 할 수 있다.