핵심 기술

소개서

|  |  |  |  |
| --- | --- | --- | --- |
| QuadCoder | HTML5 based Educational Screen Sharing System | | 정인환 교수 |
| 국남규 | 0792049 | [3gguild@gmail.com](mailto:3gguild@gmail.com) | 010-3707-3627 |
| 송치호 | 0792067 | [ikari0331@hanmail.net](mailto:ikari0331@hanmail.net) | 010-7196-2137 |
| 김민수 | 0892049 | [karosis28@gmail.com](mailto:karosis28@gmail.com) | 010-2729-2635 |
| 강지수 | 0992050 | [km3019@naver.com](mailto:km3019@naver.com) | 010-9579-1236 |
| Homepage : https://github.com/quadcoder | | | |

우리 팀이 이번 설계 프로젝트에 사용할 핵심 기술은 바로 HTML5입니다. 이는 HTML과 CSS, Javascript로 이루어진 플랫폼이며 HTML5소개와 Javascript 중에 사용하게될 Node.js에 대해서 설명하도록 하겠습니다.

* **HTML5의 개념 및 특징**

’14년에 차세대 웹 표준으로 확정 예정이며, 기존 텍스트와 하이퍼링크만 표시하던 HTML이 멀티미디어 등 다양한 애플리케이션까지 표현·제공하도록 진화한 “웹 프로그래밍 언어“입니다. 예로, 오디오·비디오·그래픽 처리, 위치정보 제공 등 다양한 기능을 제공함으로써, 웹 자체에서 처리할 수 있는 기능이 대폭 향상 되었습니다.

※ HTML(Hyper Text Markup Language)는 웹 문서를 만들 때 사용하는 프로그래밍 언어입니다.

* **HTML5의 주요기능**
* Video & AudioVideo & Audio : 비디오 및 오디오 기능을 자체적으로 지원
* 3D, Graphics & Effects3D, Graphics & Effects : 다양한 2차원 및 3차원 그래픽 기능을 지원
* Offline & StorageOffline & Storage : 네트워크 미지원 환경에서도 웹 이용을 가능하게 함
* Device AccessDevice Access : 카메라, 동작센서 등의 H/W 기능을 웹에서 직접적으로 제어
* Web SocketWeb Socket : 웹 (클라이언트)에서 서버 측과 직접적인 양방향 통신 가능
* Geo-LocationGeo-Location : GPS 없이도 단말기의 지리적인 위치 정보를 제공
* Styling EffectsStyling Effects : 글씨체, 색상, 배경 등 다양한 스타일 및 이펙트 기능 제공
* SemanticsSemantics : 웹 자료에 의미를 부여하여 사용자 의도에 맞는 맞춤형 검색 제공
* **HTML5 시장 전망**

HTML5는 2014년에 국제 표준이 확정될 것으로 예상되며, 현재 웹 브라우저 기업을 비롯한 글로벌 업체들은 오디오·비디오·그래픽 등 표준화가 많이 진행된 기능을 중심으로 HTML5를 적극적으로 도입하고 있습니다.

**1. 모바일 단말**

미국 시장조사기관인 Strategy Analytics는 전 세계적으로 HTML5를 지원하는 모바일 단말이 2009년 3백만대에서 2016년 15억 6,000만대 (전체 단말 중 85%)로 크게 늘어날 것으로 전망 하였습니다.

※ ABI Research는 HTML5 지원 모바일 단말이 2016년 21억대 이상이 될 것 이라고 긍정적으로 예측 하였습니다.

**2. 모바일 웹/앱**

모바일 애플리케이션 시장은 세계는 2011년 56억불에서 2015년 375억불로, 국내는 2011년 1.4조원에서 2015년 3.5조원 으로 성장할 것이며, 2017년에는 HTML5 기반의 모바일 웹이 모바일 앱을 넘어설 전망 (ETRI) 하였습니다.

※ 현재, 모바일 웹은 앱에 비해 기능이 떨어지나, 향후 성능이 개선될 것으로 예상 하였습니다.

* **HTML5 표준화 전망**

2004년부터 W3C(World Wide Web Consortium, 웹 표준의 제정 등을 위해 1994년 설립)를 중심으로 HTML5 표준 개발이 시작되었으며, 현재 기능별 표준화가 진행中이며, 2014년까지 최종 표준이 확정될 예정입니다.

* **표준 웹 환경의 확산**

현재, 멀티미디어를 비롯한 확장 기능들을 지원하기 위해서 이루어지고 있는 비표준 인터넷 웹 환경(ActiveX, Flash, 실버라이트 등 별도 프로그램 설치)이 점차 해소될 전망입니다.

* **개방형 생태계로의 변화**

개방된 웹上에서 다양한 애플리케이션을 구현하고, 이를 누구나 브라우저로 접근할 수 있으므로, 애플(iOS) 및 구글(안드로이드) 등 OS 플랫폼에 대한 의존이 감소될 수 있습니다.

* **이용자 선택권의 강화(One Source Multi Use)**

이용자의 경우, 인터넷에만 접속하면 스마트 폰·태블릿 PC·PC 등 기기는 물론, 애플이나 구글 등 벤더에 상관없이 S/W나 콘텐츠 등을 이용 가능합니다.

※ 현재, 애플 앱스토어에서 구입한 앱은 구글 안드로이드 폰이나 태블릿 PC에서 사용할 수 없으나, HTML5 기반의 웹으로 전환하여 이용자 선택 上의 제약 탈피가 가능함

* **자바스크립트와 V8**

오랜 시간 동안 자바스크립트(Javascript)는 많은 개발자들에게 등한시되거나 오해를 받아왔습니다. 그러나 2008년 9월, 구글이 크롬(chrome) 웹 브라우저에 이식한 자바스크립트 엔진 V8을 오픈 소스로 개방함으로써 자바스크리트에 대한 성능 문제는 사라지게 되었으며, 자바스크립트는 재조명받기 시작하였습니다. 마침내 2009년, 유럽 JSConf의 라이언 달(Ryan Dahl)이 V8을 이용하여 자바스크립트의 강력함과 단순함을 활용한 이벤트 기반의 논블로킹 I/O를 주 컨셉으로 만든 백엔드(Back-end) 자바스크립트 기반 플랫폼을 발표하였습니다. 이것이 현재 실리콘밸리에서 가장 뜨거운 감자 중 하나인 Node.js입니다.

“Node.js는 크롬 자바스크립트 엔진인 V8 기반으로 만들어진 플랫폼이며, 빠르고 확장 가능한 네트워크 프로그램을 쉽게 작성할 수 있게 합니다. 특히, 이벤트 주도(event-driven), r(non-blocking) I/O 모델을 사용함으로써, 분산 환경에서 실행되는 가장 가볍고 효과적인 그리고 완벽한 데이터 집중적(data-intensive)인 실시간 어플리케이션을 작성할 수 있게 해줍니다.”

위의 Node.js의 정의는 Node.js 공식 홈페이지(<http://nodejs.org>)에서 발췌한 내용입니다. 위의 글을 처음 접했을 때 혹자는 매우 이해하기 쉬운 용어만 나열했다고 생각할 수 있지만, 필자는 다소 낯선 단어들이 섞여 있었다고 느꼇습니다. Node.js를 본격적으로 이해하기에 앞서 기본적으로 알아야 할 내용에 대해 짚고 넘어가겠습니다.

* **자바스크립트란**

자바스크립트는 웹 브라우저에서 실행되는 스크립트 언어로, HTML로 작성된 웹페이지에 함수 형태로 작성되는 언어입니다.

간혹 자바와 자바스크립트가 어떤 연관성이 있는지 궁금해하는 분들이 있습니다. 자바와 자바스크립트의 비슷한 점은 두 언어 모두 C 언어와 비슷한 구문을 사용하고, 객체지향 언어이며 자바스크립트를 설계할 때 자바의 구문과 표준 라이브러리를 참조하여 설계하였다는 점입니다. 대표적인 예를 들자면 모든 자바의 예약어는 자바스크립트에서도 예약어로 설정되어 있고 작명 규칙, Math, Date 객체는 자바 1.0을 기초로 작성되어 있습니다.

두 언어의 차이점은 자바스크립트는 해석형 언어이고 컴파일할 필요 없이 실행 시에 사람이 읽을 수 있는 소스코드로부터 해석되고, 자바는 컴파일을 반드시 해야 하며, 컴파일된 바이트 코드에 의해 실행된다는 점입니다. 그리고 데이터 타입 처리 방식도 서로 다릅니다. 자바스크립트는 느슨한 타입(week typing)이므로 데이터 타입에 크게 민감하지 않습니다. 가령, 숫자 2와 문자 2를 서로 붙이거나 더할 수 있지만 (2+’2’=4) 자바는 강한 타입(strong type)이므로 숫자 2와 문자 2를 서로 붙이거나 더하면 컴파일할 때 바로 에러가 나타나므로 타입 캐스팅을 해야 합니다.

위에서 잠시 언급했던 해석형 언어(Interpreted Language)란 컴파일할 필요 없이 소스 코드 자체가 실행시에 인터프리터에 의해 해석되어지는 언어를 의미합니다. 일반적으로는 자바와 같이 컴파일된 바이트 코드를 실행시키는 것이 속도가 빠르지만, 개발하는 관점에서 보면 그저 빠르다고 볼 수 없습니다. 왜냐하면 소스에 변경사항이 생긴 경우에 시간 및 리소스를 들여 재 컴파일하는 작업이 필요하기 떄문입니다.

물론, 소규모 프로젝트인 경우에는 시간이 얼마 걸리지 않겠지만 엔터프라이즈급인 경우에는 어떨까요? 자바 기반의 엔터프라이즈급 프로젝트 개발 환경을 소화하기 위해서는 일반적인 노트북의 사양(32bit O/S, 4GB 메모리, HDD)으로는 턱없이 부족합니다. 단순히 소스를 빌드하여 개인 개발 환경에 탑재하는데 걸리는 시간은 무시하지 못 하는 수준입니다. 하드디스크를 SDD로 바꾼 이후에 어느 정도 견딜만하지만, 그렇지 못한 환경에서는 생산성을 떨어뜨리는 주요인이 됩니다. 또한, 운영 환경이 릴리즈하는 주기에도 큰 영향을 미칩니다. 이와 같은 이유로 많은 프로젝트에서 해석형 언어를 주로 사용합니다. 그렇다 보니 자바스크립트를 사용할 때에는 어떤 해석 엔진을 사용하느냐가 성능 상에 큰 이슈가 되어 왔으며, 이 이율 백엔드 프로그램에서의 사용을 기피했는지도 모르겠습니다. 하지만 2008년 9월, 성능 이슈 및 엔진에 대한 문제점을 해결해 준 녀석이 등장했습니다. 바로 V8 입니다.

* **Node.js의 심장, 크롬 자바스크립트 실행 엔진 V8**

V8 자바스크립트 엔진은 구글에 의해서 C++로 개발된 오픈 소스 자바스크립트 엔진으로 2008년 9월 2일, 구글의 웹 브라우저인 크롬에 내장되어 공개되었습니다.

V8은 기존의 컴파일된 바이트 코드를 실행시키거나 인터프리터로 해석하는 대신에 자바스크립트를 컴파일하여 네이티브 머신 코드로 변경한 후에 실행할 수 있게 해줍니다. 이렇게 컴파일된 코드는 코드의 실행 프로파일을 분석하여, 실행 시에 다시 한 번 최적화됩니다. V8은 객체에 할당되는 메모리를 효율적으로 관리하며, 더 이상 필요없는 객체는 수집하여 버립니다(garbage collection). 정확한 가비지 콜렉터는 V8의 고성능에 이바지하는 주요한 키 포인트입니다. 이러한 V8 엔진은 웹 브라우저(크롬)에서 실행될 뿐 아니라 독립형(standalone)의 고성능 엔진으로 사용할 수 있고 웹 브라우저와는 전혀 상관없이 사용할 수 있습니다. 웹 브라우저 없이 구동되는 자바스크립트는 조금 생소하게 느껴질 수도 있을 것입니다. 이 훌륭한 V8 엔진을 탑재하여 독립형 자바스크립트 언어로 개발을 가능하게 한 플랫폼이 바로 Node.js입니다.

Node.js를 사용하면 더 이상 HTML 상단에 자바스크립트를 선언하여 호출하는 방법으로 ‘삽입’하여 사용할 필요가 없습니다. 자바스크립트로만 작성된 코드를 손쉽게 구동할 수 있습니다. 자바스크립트는 그 자체가 ‘완전한’ 언어이며 Node.js를 통해 이를 십분 활용할 수 있습니다. V8 엔진에 대하여 더 자세한 사항을 알고 싶다면 다음 링크를 참고하기 바랍니다.

<http://developers.google.com/v8/intro>

* **Node.js 이해하기**

Node.js의 태생에 대해 알아보았으므로 이제 Node.js의 설계 사상과 컨셉에 대해 알아보겠습니다. 서두의 Node.js는 이벤트 주도 및 논블로킹 I/O 모델을 사용하여, 무척 효율적인 분산 환경의 네트워크 프로그램이 가능하다고 언급하였습니다. 여기에서는 이벤트 주도, 논블로킹 I/O에 대해 설명하겠습니다.

대형 할인 매장을 가면 사고 싶은 물건을 모두 선택한 다음 계산을 하기 위해 계산대 앞에 줄을 섭니다. 평일 낮 시간이라 계산하는 점원이 1며밖에 없고 3명의 손님이 차례를 기다리고 있다고 가정해 보겠습니다. 3명의 손님이 모두 계산할 때까지 기다려야 합니다. 1명의 손님이 계산이 끝나는데 평균 1분 정도소요된다고 하면 3분 동안 줄이 줄어들길 기다려야 합니다. 이것을 바로 블로키(blocking)이라고 합니다.

이번에는 갑자기 매장에 손님이 많아졌습니다. 계산하는 점원이 1명 더 왔으므로 앞에 기다리던 손님이 옆의 계싼대로 옮겨가면서 기다리던 줄이 짧아졌습니다. 동시에 점원 2명이 계싼하기 때문입니다. 하지만, 아직도 앞의 손님이 계산하는 동안 기다려야합니다. 이 예는 분산 병렬 처리를 위한 좋은 예가 됩니다. 만약 자바로 프로그래밍을 한다면 현재 카운터에서 계산하는 점원들을 Runnable 구현체로 만들어, 멀티 쓰레딩 프로그램을 만들 것입니다. 하지만 기다리는 시간을 어떻게든 줄이고 싶습니다. 여기 서 있는시간은 득이 될것이 전혀 없습니다. 시스템 역시 그렇습니다. 실제로 데이터를 처리하는순간 이외에 블로킹되는 시간동안 자원은 할당되어 있으나 실질적으로는 사용하지 않는 비효율적인 부분이 생기게 되었습니다. 어떻게 이 부분을 개선할 수 있을까요?

다른 예를 하나 들어보겠습니다. 주말에 커피숍은 언제나 그렇듯이 사람들로 북적거립니다. 도착하자마자 카운터에 가서 커피를 주문하고 동그란 무선 호출기를 받습니다. 커피가 준비되면 알람을 울려 커피가 준비되었음을 알려주는 호출기입니다. 호출기를 받고, 구석진 자리를 선택해 노트북의 전원 플러그를 꼽을 수 있는지 확인합니다. 노트북을 세팅하고, 가져온 참고 문서를 꺼내고, MP3 플레이어 및 웹 서핑용도의 태블릿 PC를 노트북 옆에 세워 놓았습니다. 그리고 창 밖의 멋진 풍경을 잠시 감상하였습니다. 갑자기 호출기가 붉은 빛을 토하면서 울립니다. 아까 주문한 아이스 더치가 준비되었나 봅니다. 카운터로 가서 주문했던 커피를 받아서 다시 자리에 앉았습니다.

커피를 주문한 다음 준비가 될 때까지 전혀 간섭받지 않고 하고 싶은 일을 수행했습니다. 블로킹을 당하지 않았습니다(non-blocking). 카운터의 종업원은 필자가 주문하는 이벤트, 그리고 커피 준비가 완료되었다는 두 가지 이벤트를 받았습니다. 종업원은 해당 이벤트가 발생했을 때 본인이 해야 할 일을 했습니다. 이것이 바로 이벤트 주도입니다. 여기서 필자가 커피가 완료되었다고 알려준 호출기가 중요한 포인트입니다. 이를 프로그래밍에서는 이벤트 콜백(evenet callback)이라고 합니다.

조금 더 이해를 돕기 위해서 소스 코드와 함께 살펴보겠습니다. 다음 예제 소스 및 설명은 Node.js의 창시자인 라이언 달의 초창기 발표 자료(2009년 11월 8일)를 참고하였습니다. 많은 웹 어플리케이션이 다음과 같은 코드를 가지고 있습니다.

|  |
| --- |
| var result = db.query(“select \* from T”);  //use result |

쿼리를 수행한 다음 받은 결과를 가지고 다음 단계를 수행합니다. 쿼리를 수행합니다. 쿼리를 수행하는 동안 어플리케이션은 무엇을 할까요? 대부분의 경우 쿼리의 응답을 그저 기다리기만 합니다. 앞서 언급한 것처럼 멀티쓰레딩 등을 통해 멀티태스킹을 한다면 좀 더 나아지겠지요? 최소한 특정 쓰레드의 대기 시간 동안 다른 쓰레드들이 수행되기 때문입니다.

위의 소스 코드를 다음과 같이 바꿔 보겠습니다.

|  |
| --- |
| db.query(“select \* from T”, function (result){  //use result  }); |

query 함수의 두 번째 인자값에 콜백 함수를 선언하여 결과를 어떻게 사용할지 구현하였습니다. query 함수 호출 시, 결과에 따른 후행 작업을 미리 선언하여 인자값으로 넘겨 준 것입니다. 이런 프로그래밍 방식을 이벤트 주도 프로그래밍이라고 하며, Node.js의 대표적인 특징 중 하나이다. 여러 개의 I/O 작업 요청이 들어오면 병렬적으로 I/O 작업을 수행하고, 완료 시 각각의 콜백 함수를 실행합니다.

그러면 이벤트를 관리하는 녀석이 있겠죠? 이 녀석을 ‘이벤트 핸들러’라고 합니다. 이벤트 핸들러는 이벤트 루프를 무한 반복 수행합니다. 이벤트 루프의 주된 역할은 이벤트 감지와 이벤트 핸들러 트리거(수행자) 두 가지 입니다. 이벤트 핸들러는 단일 쓰레드로 실행되며 어떤 이벤트가 발생하더라고 방해받지 않고 자기 역할을 수행합니다. 이런 특징 때문에, 프로그래머는 동기화 요구 조건이나 공유 메모리 상태 변경등에 대한 다중 쓰레딩 관련 부분에 대해 신경 쓰지 않아도 됩니다.

자바스크립트는 비동기 함수, 클로저, 콜백 함수 등의 특징으로 이벤트 루프와 함께 사용하기 위해 특별하게 설계되었습니다. 이게 바로 Node.js가 자바스크립트 기반으로 작성된 이유이며 이벤트 기반, 논블로킹 I/O 모델 사용을 통해 고성능, 동시다발적으로 수행되는 프로그램을 작성할 수 있는 이유입니다.

아래 이미지는 지금까지 설명한 Node.js의 동작 원리를 잘 표현하고 있습니다.

|  |
| --- |
|  |

* **Node.js의 아키텍처 및 동작 원리**

Node.js에 대하여 어느 정도 이해되었을 것 입니다. 이제 조금만 더 껍집을 벗겨보도록 하겠습니다. Node.js의 아키텍처를 살펴보고 실제로 어떻게 동작하는지 알아 보겠습니다.

Node.js의 아키텍처는 어떻게 구성되어져 있을까요? 이번에도 위에서 참조하였던 라이언 달의 동영상의 일부를 참고해봅시다.

|  |
| --- |
| * Node.js 내부의 아키텍처 구성도 |

위의 그림은 Node.js 내부의 아키텍처 구성도입니다. 주요 컴포넌트에 대한 설명은 다음과 같습니다.

* V8 : 구글의 자바스크립트 엔진
* libev : 이벤트 루프에서 사용하는 라이브러리
* libeio : 쓰레드 풀 라이브러리
* http-parser : http 분석을 하는 파서
* evcom : 스트림 소켓 라이브러리로 libev의 맨 위에 위치
* udns : non-blocking DNS resolver
* **Node.js 적용 사례**

Node.js는 아직 베타 버전이지만 수많은 기업들에 의해 채택되어져 운영 환경에서 사용하고 있습니다. 왜 그들은 Node.js를 선택해야만 했을까요? 물론, 여러 가지 원인이 있겠지만 여러 사례들을 보았을 때 다음과 같은 공통된 이유가 있었습니다.

* 전 세계적으로 모바일을 포함한 다양한 장치를 통해서 동시에 접속하는 대량 트랜잭션에도 빠르고 안정적인 처리량 및 응답 속도를 제공해야 하는 경우
* 제한된 물리적인 리소스에도 대량 처리량을 지속적으로 처리해야 하는 경우
* 개발 플랫폼을 변경해야 하는 시점에 개발자들이 이미 알고 있는 개발 언어인 자바스크립트를 사용함으로써 Learing Curve를 최소화해야 하는 경우
* 프론트엔드(Front-end)의 프론트엔드 소스와 백엔드(Back-end)의 백엔드 소스를 한 가지 언어로 통일시킴으로써 전사 표준 개발 프레임워크를 클라이언트/서버 구분 없이 단순하게 가져가고, 개발 조직 역시 구분하지 않게 프로젝트를 수행해야 하는 경우

이외에도 여러 가지 세부 원인들이 있습니다. 이런 사항들을 실제 적용 사례를 통해 현재 어떤 기업들이 Node.js를 어떻게 활용하고 있는지 알아보고, 미래 모습에 대해서도 생각하는 시간을 가져보겠습니다.

1. **이베이의 현명한 선택, Node.js**

이베이(<http://www.ebay.com/>)는 전 세계 30여 개의 국가에서 수십 억 달러의 거래가 오고 가는 개인 간의 거래를 주선해주는 사이트입니다. 이베이는 다음과 같은 이유로 차세대 웹 어플리케이션의 소프트웨어 스택을 선정하는데 고민합니다.

* 노트북, PC, 스마트폰을 비롯한 다양한 매체에서 접근이 가능한 웹어플리케이션
* 다양한 매체는 데이터의 Input/Output이 가능해야 합니다
* 어플리케이션들은 물리적으로 서로 다른 서버에서 다른 서비스들을 제공하고 있습니다
* 어플리케이션들은 다양한 3rd-party 데이터 소스들을 참조합니다
* 클라이언트-서버 간 실시간 양방향 통신이 가능해야 합니다

이베이는 잘 알려진 자바의 요소 기술을 응용하여 이베이 내부에서 전체 기능을 제공하게 개발하느냐 아니면 아직 잘 알려지지 않았지만 비동기식 I/O를 훌륭하게 제공하는 Node.js를 사용하느냐라는 두 가지 방향에 대하여 결정해야 했습니다. 그 당시만 해도 Node.js에 대한 경험을 가지고 있는 회사가 몇 개 되지 않았으므로 결정하는 것이 쉽지 않았습니다. 이베이는 전체 시스템에 미치는 다음 항목들을 가지고, 상세 비교 분석을 통해 플랫폼을 선정했습니다.

* I/O 부하 상황에 대한 성능 및 확장성 : 자바스크립트의 수행은 부하를 초래하긴 하지만 중요한건 I/O 바운드입니다. 블로킹 I/O는 부하를 해결하는데 전혀 도움이 되지 않습니다.
* 사용 편의성 : 실시간으로 모니터링이 가능해야 하며, 장애 상황 시에 빠르게 대응해야 합니다. 신기술 도입 시, 이베이의 로깅 및 모니터링 툴과의 가용 여부는 반드시 가능해야 합니다.
* 각 커넥션에 최소 메모리 할당 여부 : 자바스크립트 수행 시 경우에 따라 느린 혹은 빠른 API를 포함하고 있을 수 있으나, 중요한 것은 개방된 커넥션 개수가 증가하더라도 안정적으로 서비스를 제공할 수 있는 것입니다.
* 동적 언어 제공 : 최소 인원으로 빠르게 빌드를 해야 하고, 새로운 유스케이스에도 버그를 쉽게 발견하고, 신속하게 이터레이션을 수행할 수 있어야 합니다. 또한, 개발자가 적은 코드로도 쉽게 확장 할 수 있어야 합니다.

결국, 이베이는 자바스크립트를 개발 언어로 선정하고 Node.js를 런타임 스택으로 결정합니다. 이를 통해 이베이가 얻은 것은 다음과 같습니다.

* 자바스크립트와 Node.js는 개발 주기를 무척 짧게 해주었습니다. 비록, 최초 선정 시 바른 툴 및 라이브러리 선정에 걱정이 많았지만 Node.js 에코시스템은 복잡한 시스템을 구축하는데 명확한 방안을 제시해주었습니다.
* 우분투 워크스테이션에서 일반 개발자가 제어할 수 있는 커넥션의 개수가 각 Node.js 프로세스 당 12만 여 개로 증가하였으며, 각 커넥션은 단지 2KB의 메모리만을 소비하게 되었습니다. 이는 더 많은 커넥션을 제어할 수 있다는 것을 알게 해 주었으며, Node.js를 선택하여 사용하는데 자신감을 불어넣어 주었습니다.
* Node.js의 이벤트 기반 비동기식 I/O는 다중쓰레드 비동기식 I/O 프로그램 구현 시 항상 겪었던 블로킹과 데이터 동시 실행 관련 문제를 고민하지 않게 해주었습니다.

이렇게 구축된 이베이만의 개발 프레임워크가 바로 ql.io(<http://ql.io)>입니다. ql.io에 대한 소개 및 Node.js를 선정하게 된 배경에 대해 더 관심이 있다면 다음 링크를 확인해보기 바랍니다.

* <http://www.ebaytechblog.com/2011/11/30/announcing-ql-io>)

1. **왜 링크드인은 Node.js를 선택했나**

링크드인(www.linkedin.com)은 200여 개의 국가에서 약 20억 명의 회원 수를 보유하고 있는 전세계에서 가장 큰 전문가들의 소셜 네트워크 서비스입니다. 링크드인은 현재 모바일 장치들과 REST를 통신하기 위한 모바일 서버를 기존 Ruby on Rails에서 Node.js로 모두 재작성하여 사용 중입니다. 다음 그림은 링크드인의 모바일 서버 아키텍처의 구성입니다.

|  |
| --- |
| * 링크드인 모바일 서버 아키텍처 구성 |

이를 통해서 링크드인이 얻은 것은 무엇일까요? 다음 항목을 통해 링크드인이 왜 Node.js를 선택했는지 알 수 있습니다.

* 성능이 향상되었으며, 기존에 비해 적은 메모리를 사용합니다. 어떤 경우에는 무려 20배나 빠른 응답 속도를 보여줍니다.
* 프로그래머들은 이미 가지고 있던 자바스크립트 기술을 활용할 수 있습니다. 다른 언어를 굳이 배울 필요가 없습니다. (대부분의 개발자들은 자바스크립트에 대한 경험이 있기 때문입니다.
* 프론트엔드(Front-end)와 백엔드(Back-end) 개발 팀이 한 개의 팀으로 통합되었습니다. 이는 동일한 개발 플랫폼을 사용함으로써 얻은 큰 이점입니다.
* 서버의 CPU 사용량이 30% 수준에서 3%대로 떨어졌습니다. 무려 10배의 리소스 최적화 상황을 보여주고 있습니다.
* 개발자들은 개발에만 더 집중하게 되었습니다. 가령, 개발 환경을 구성하기 위해 운영 환경에서 사용하는 WAS를 본인의 랩탑(laptop)에 설치하고, 빌드 및 테스트 환경을 구성하고, 문제가 생길 때마다 반복적으로 확인하는 일 등이 사라졌습니다.

1. **야후의 개발자들을 위한 전략**

야후([www.yahoo,com](http://www.yahoo,com))는 미국에서 가장 유명한 검색 포탈 서비스 중 하나입니다. 야후는 2010년 초부터 Node.js에 관심을 가지고 있었습니다. 야후의 플랫폼 부사장인 Bruno Fernandez-Ruiz는 그 당시에 Node.js를 단순히 파일을 업로드하는 정도로 사용했다고 합니다. 그들은 프론트엔드와 백엔드의 프로그램 언어를 단일 언어로 통일시키고 싶었는데, Node.js를 사용하는 것이 유일한 방법이었습니다. 결국 그들은 12개월을 투자하여 자신들만의 개발 환경 및 플랫폼을 구축하였습니다. 이 프로젝트명을 ‘칵테일즈(Cocktails)’라 명명하였습니다. 칵테일즈는 Node.js뿐 만 아니라 HTML5, CSS3, 자바스크립트 등이 섞여 있었으며 2011년 말에 Node.js와 관련 있는 2개의 칵테일즈를 발표했습니다.

첫 번째 칵테일은 Mojito입니다. 자바스크립트와 Node.js를 통해 클라이언트 및 백엔드 프로그래밍을 동일 코드로 작성할 수 있게 해줍니다. 이렇게 작성된 배포할 수 있는 호스트 환경이 바로 또 하나의 칵테일 ‘Manhattan’입니다. 다음 그림은 칵테일즈를 설명한 그림입니다.

|  |
| --- |
| * 야후 칵테일즈에 대한 소개 |

1. **야머의 Node.js 구현**

야머([www.yammer.com](http://www.yammer.com))는 전 세계의 20만여 개의 회사가 사용하고 있는 Private 소셜 네트워크 서비스입니다. 야머는 여러 운영 환경에서 Node.js를 활용하고 있습니다. 가령, 야머로 업로드하는 모든 파일은 Node.js서버를 통해 저장됩니다. 야머의 공동 작업이 가능한, 문서 작성 서비스 역시 Node.js로 구현되어 있습니다. 이외에 야머의 내부 시스템에서도 Node.js를 활용하고 있습니다. 빌드 시스템이나 개발자 프록시 셋업 프로그램 등입니다.

야머는 Node.js를 통해 무엇으 얻었을까요>

* Node.js는 확장성이 좋습니다. 야머는 끊임없이 서비스가 중단되는 현상에 대하여 걱정할 필요없이, 손쉽고 빠르게 웹 서비스를 확장시킬 수 있습니다.
* Node.js는 개발 생산성이 뛰어나고 쉽게 새 기능을 추가할 수 있습니다. 수정된 소스를 지속적으로 통합하고 배포하는 방법이 단순하여 릴리즈 주기를 짧게 할 수 있습니다.
* Node.js는 생기가 넘치는 커뮤니티를 가지고 있습니다. 아무도 이해하지 못하는 문제점을 새벽 3시에 지문하더라도, Node.js를 사용하고 있는 개발자들의 훌륭한 커뮤니티를 통해 바로 답을 얻을 수 있는 수준입니다.
* **이번 설계에 왜 Node.js를 사용하는가?**

앞서 작성했던 내용을 보면 Node.js는 웹에서의 확실한 트렌드 기술로 자리 잡혀가고 있습니다. 저희는 여러 서버 프로그래밍 방법 중에 Node.js를 사용함으로써 트렌드 기술을 습득하는 목표를 갖고 있습니다. 또한, 서버 프로그래밍을 공부하여 기존 프로그래밍 지식에서 한발 더 앞서 나아가는 기회가 되리라 생각하고 있습니다.