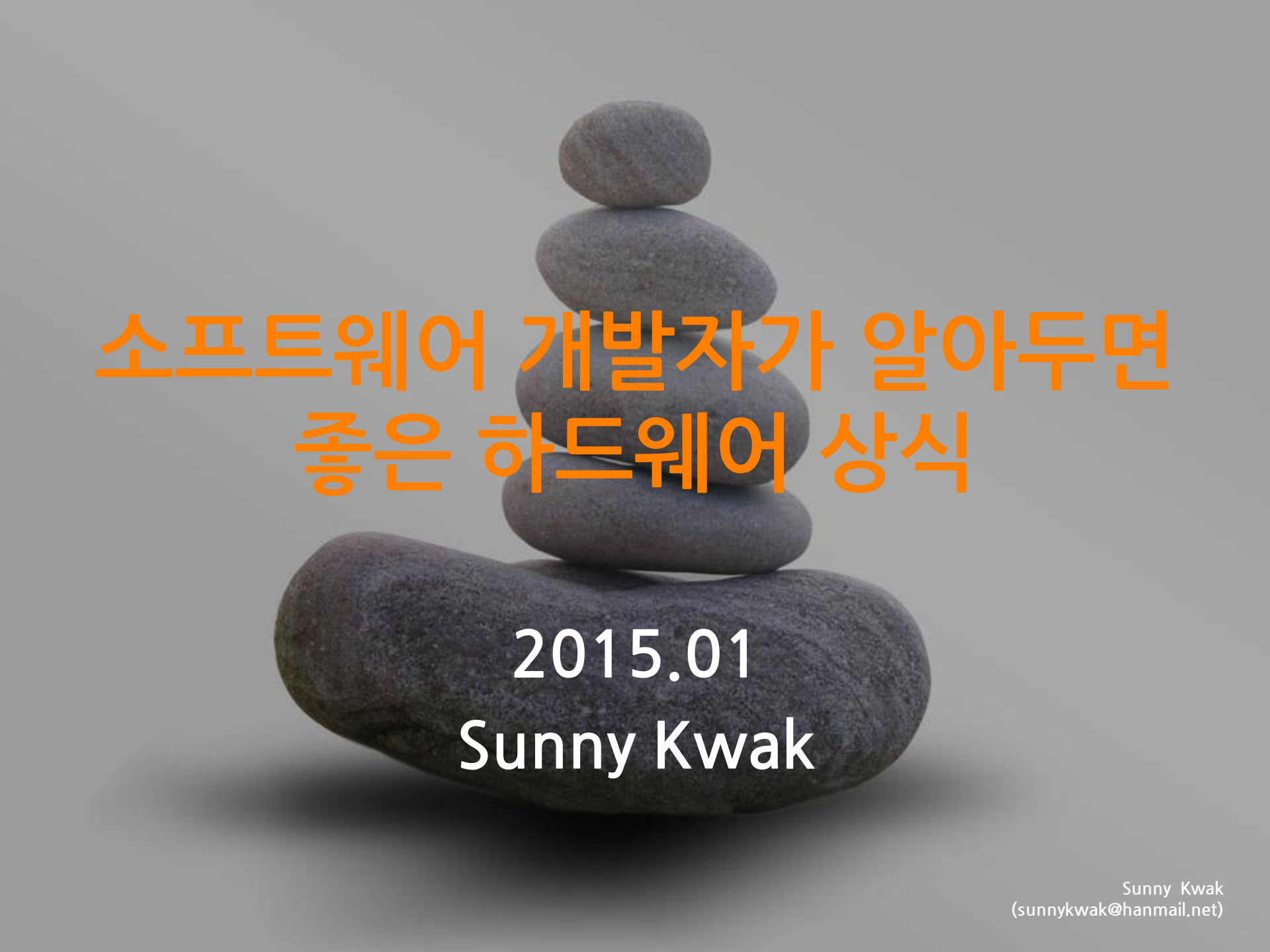


소프트웨어 개발자가 알아두면 좋은 하드웨어 상식

2015.01

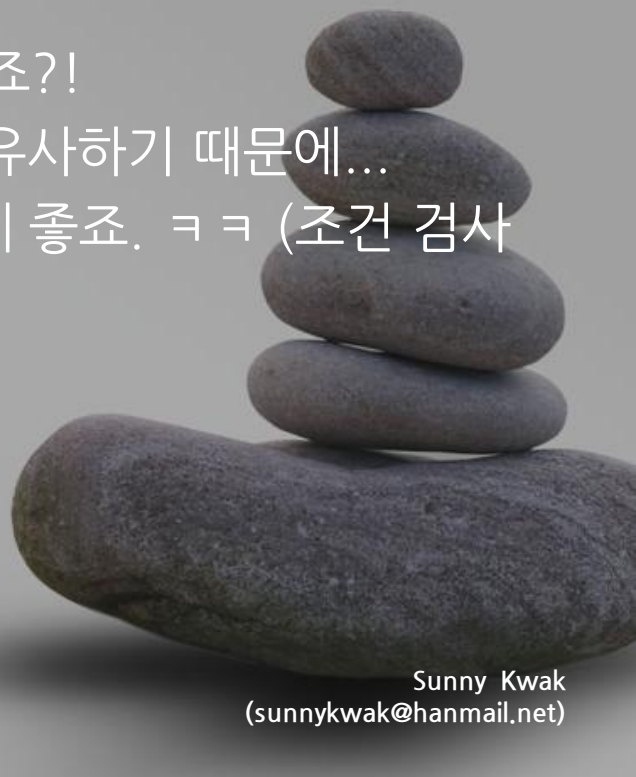
Sunny Kwak

Sunny Kwak
(sunnykwak@hanmail.net)

2014년 연말, 개발자들의 대화

- 어느 날, 페이스북 “생활코딩” 그룹에 올라온 질문
  - “if를 쓰실 때 보통 ==를 먼저 쓰시나요 !=를 먼저 쓰시나요”

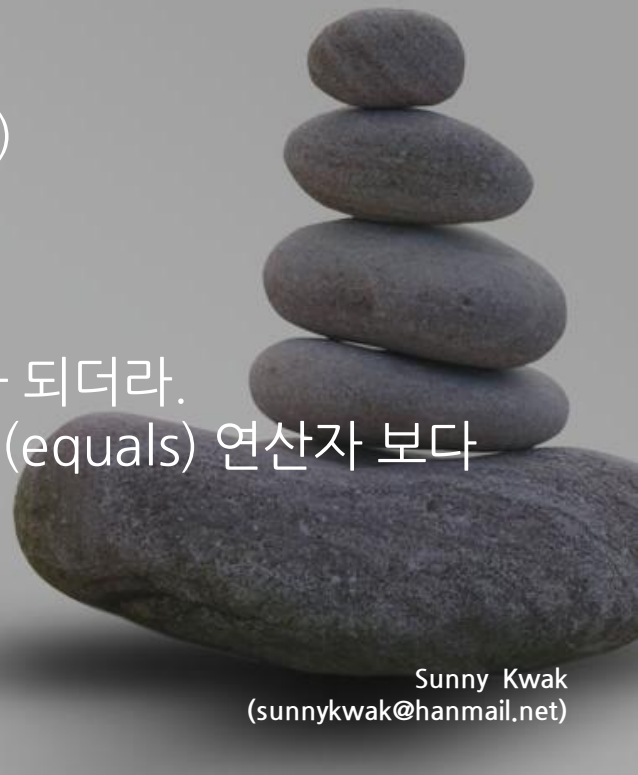
- 그리고, 이어지는 다양한 댓글들…
  - === 도 있습니다만, 취향이니 존중해 주시죠?!
  - ==를 쓰는게… 그것이 인간의 사고방식과 유사하기 때문에…
  - False match 확률이 가장 높은 놈을 넣는 게 좋죠. ㅋㅋ (조건 검사 횟수를 줄여서 성능을 향상 시킬 수 있다.)
  - ‘변수 == 상수’ vs ‘상수 == 변수’도 있죠.
    가독성은 전자, 실수방지는 후자.



Sunny Kwak
(sunnykwak@hanmail.net)

Alert! 떡밥 투척!

- 신선한 '이론'이 제시 되었습니다!
  - == 보다 !=가 더 빨라서 != 로 쓰는 것으로 알고 있습니다.
    그리고, 최근 컴파일러들은 자동으로 !=로 최적화 해줍니다.
- 레알? S/W 경력 20년에 첨듣는데?
  - 태클 걸어 죄송한데 이론의 출처 쫌... (진지)
- 출처 혹은 이론은...
  - Visual Studio에서 컴파일하면 !=로 최적화 되더라.
    그렇다는 건 != (not euqals) 연산자가 == (equals) 연산자 보다
    빠르기 때문이 아닌가? (그렇게 이해했다.)

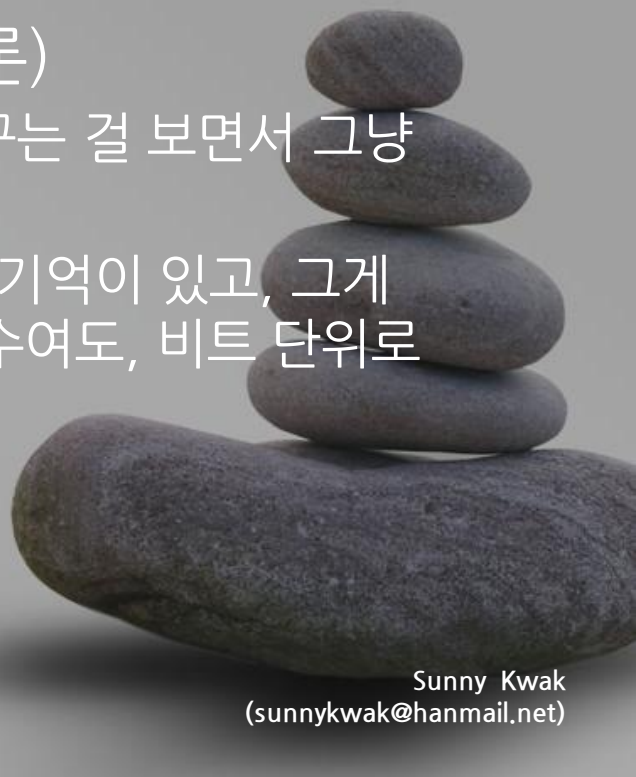


Sunny Kwak
(sunnykwak@hanmail.net)

이번 떡밥은 내가 물테야!!

- 새로운 학설에 대처하는 공학도의 자세
  - not equal 이랑 equal 명령어느 CPU 클럭 수 (속도) 차이가 나지 않는데, 전자과 학회에 질문을 들고 가야 할까요? (농담)

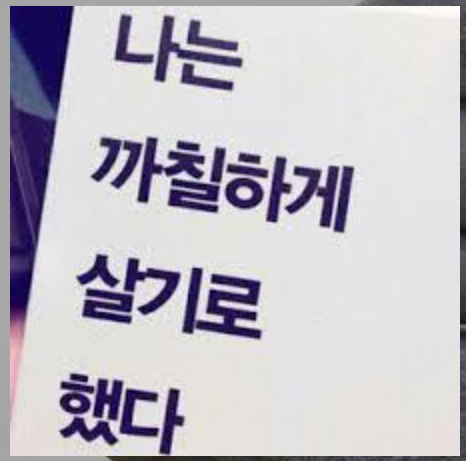
- NOT equals 연산자에 대한 경험담(이론)
  - 컴파일러 최적화를 하면 대부분이 !=로 바꾸는 걸 보면서 그냥 이게 맞나보다 하면서 코딩 습관이 생겼다.
  - '문자열 비교' 에서 != 가 빠르다는 글을 본 기억이 있고, 그게 당연히 맞다고 생각한다. 문자열이 아닌 정수여도, 비트 단위로 비교한다 생각하면 같은 이치 아닌가?



Sunny Kwak
(sunnykwak@hanmail.net)

나는 왜 떡밥을 물었는가?

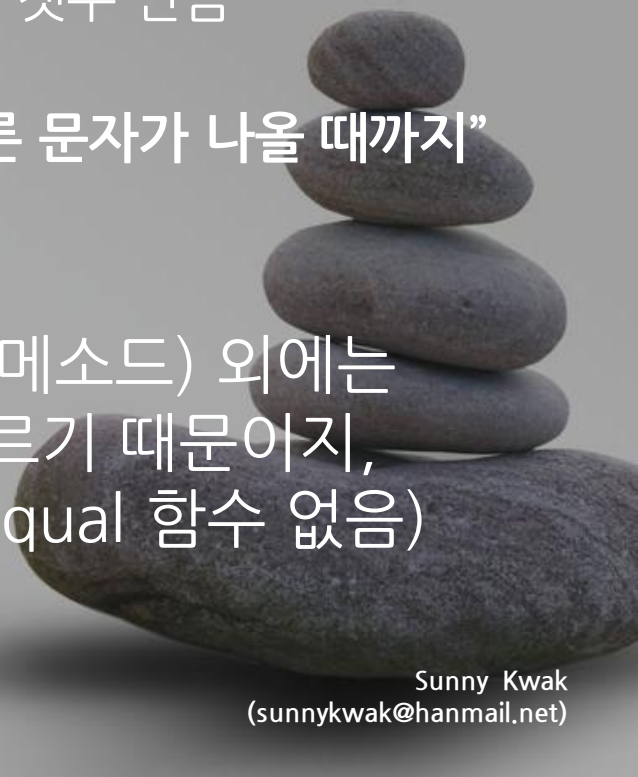
- 공학도(engineer)는 '정확한 지식'으로 무장해야 한다.
- '철저한 논리' 혹은 '확실한 근거' 없는 지식을 바탕으로 개발된 '소프트웨어'를 신뢰할 수 있는가?
- 잘못 전파된 지식은 많은 주변 사람들을 불행하게 만든다.



Sunny Kwak
(sunnykwak@hanmail.net)

무엇이 문제인가? (1)

- 문자열 2개를 비교할 때, 다른 것과 같은 것을 비교하는 경우 어느 쪽이 빠를까?
  - “abcdefg”와 “abcdefg” 가 같을 경우, 문자 갯수 만큼 **“모두“** 비교를 해야 한다. (7번 비교)
  - “abcdefg”와 “ab**Z**defg” 가 다를 경우, **“다른 문자가 나올 때까지”** 비교한다. (3번 비교)
- 정작, 문자열 비교 함수는 equals 함수(메소드) 외에는 없다. 처리 속는 비교 데이터 같거나 다르기 때문이지, 연산자가 다르기 때문이 아니다. (not equal 함수 없음)



Sunny Kwak
(sunnykwak@hanmail.net)

무엇이 문제인가? (2)

- 그렇다면 기본형(primitive type)을 비교할 때도 성능 차이가 있을까?
  - 예를 들어, int a 와 int b 변수에 들어 있는 값을 비교할 때...
  - 비트(bit) 단위로 비교하면, 문자열 비교와 같은 맥락의 성능 차이가 있을 것이다.
- 하지만, CPU는 기본형 데이터를 비교할 때, 비트(bit) 단위로 비교하지 않습니다.
- **소프트웨어 개발자가 하드웨어 원리를 몰라서 문제!**

Sunny Kwak
(sunnykwak@hanmail.net)

CPU는 어떻게 동작하는가?

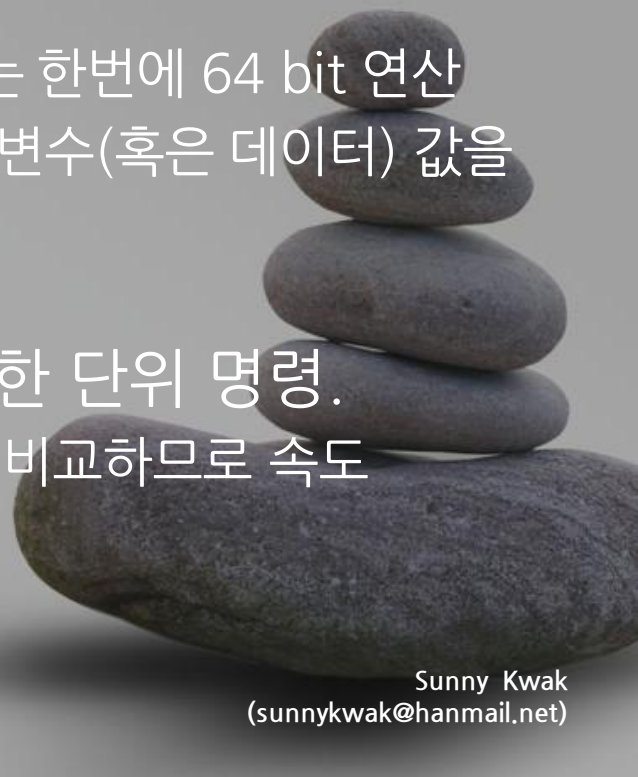
- ‘Clock’ 과 ‘ALU’
  - ‘clock’ 이 일정 주기마다 작업 신호를 ‘ALU (Arithmetic-logic Unit)’에 보내면 ALU는 한 단위의 명령을 실행한다.
  - clock은 ‘메트로놈’, ALU는 주판에 비유할 수 있다. 메트로놈이 주기적으로 움직일 때마다, 주판 알이 움직인다.

signal

Sunny Kwak
(sunnykwak@hanmail.net)

클럭 그리고 산술논리 연산

- 컴퓨터의 연산 속도는 클럭 수에 비례
  - 1 Khz = 초당 1천회, 1 Mhz = 초당 1 백만, 1 GHz = 10억
- 산술/논리 연산
  - 32 bit CPU는 한번에 32 bit, 64 bit CPU는 한번에 64 bit 연산
  - 32 bit CPU는 클럭 한 번에 4 byte 크기의 변수(혹은 데이터) 값을 비교하거나, 사칙연산으로 계산할 수 있다.
- 비교 연산은 Equals, Not equals 모두 한 단위 명령.
  - 한번에 32 bit (64 bit CPU 라면 64 bit)를 비교하므로 속도 차이는 없다.



Sunny Kwak
(sunnykwak@hanmail.net)

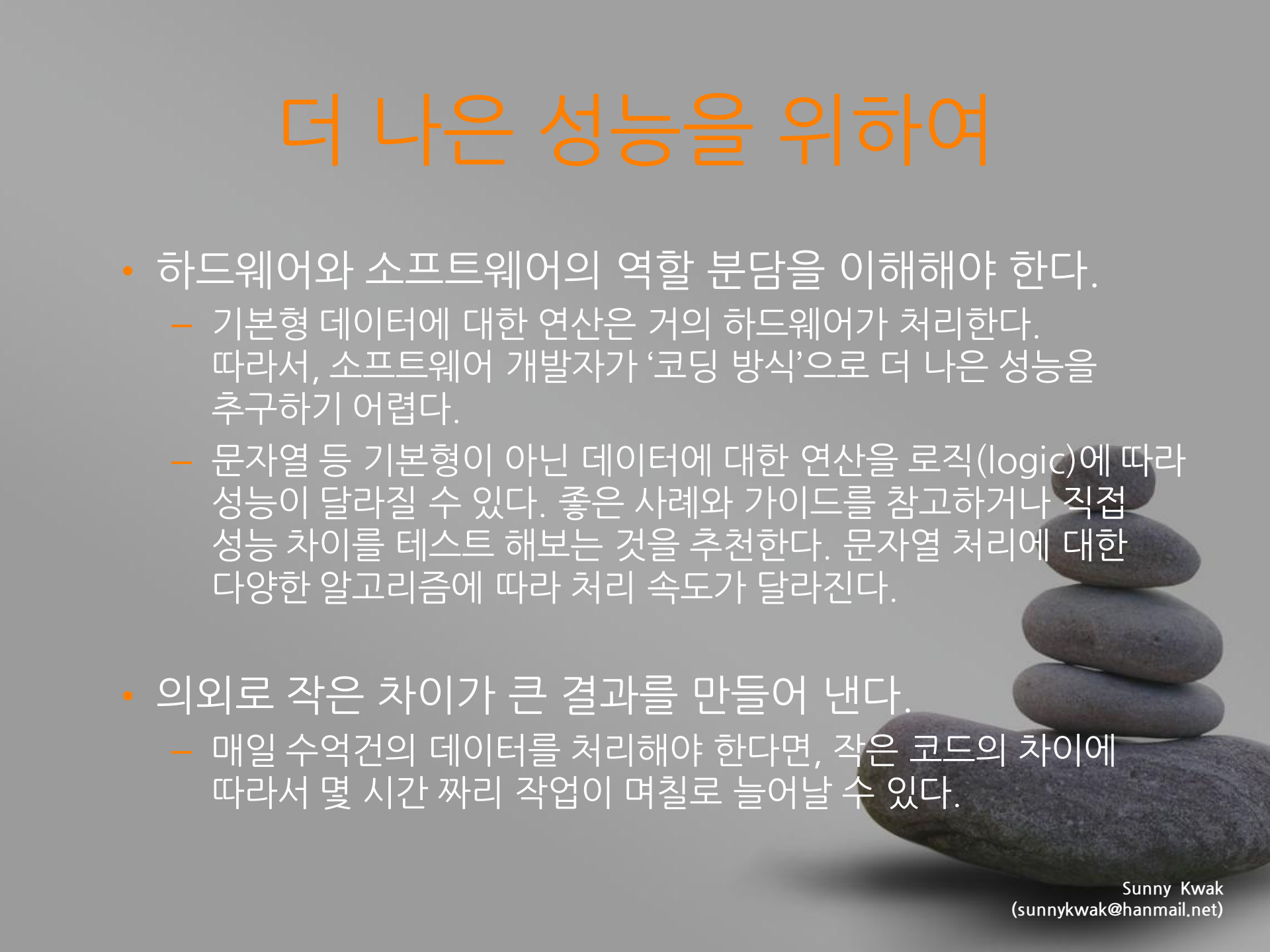
Equals vs. Not Equals

- 기본형(primitive) 타입 데이터를 비교할 때,
  - '==' 연산자와 '!=' 연산자의 속도 차이는 없다.
  - 취향, 가독성, False match 가능성 등을 고려하면 된다.
  - 정수형(int), 문자형(char), 불린형(boolean) 등이 기본형
  - 다만, 실수형(float, double)은 CPU 타입에 따라 미세한 차이가 있을 수 있다.
- 문자열 타입 데이터를 비교할 때,
  - 소프트웨어로 처리하지만, 비교 방법은 하나 뿐이다.
  - 다른 문자를 만날 때 까지, 문자를 반복(loop) 검사한다.

Sunny Kwak
(sunnykwak@hanmail.net)

더 나은 성능을 위하여

- 하드웨어와 소프트웨어의 역할 분담을 이해해야 한다.
  - 기본형 데이터에 대한 연산은 거의 하드웨어가 처리한다. 따라서, 소프트웨어 개발자가 '코딩 방식'으로 더 나은 성능을 추구하기 어렵다.
  - 문자열 등 기본형이 아닌 데이터에 대한 연산을 로직(logic)에 따라 성능이 달라질 수 있다. 좋은 사례와 가이드를 참고하거나 직접 성능 차이를 테스트 해보는 것을 추천한다. 문자열 처리에 대한 다양한 알고리즘에 따라 처리 속도가 달라진다.
- 의외로 작은 차이가 큰 결과를 만들어 낸다.
  - 매일 수억건의 데이터를 처리해야 한다면, 작은 코드의 차이에 따라서 몇 시간 짜리 작업이 며칠로 늘어날 수 있다.



Sunny Kwak
(sunnykwak@hanmail.net)

알아두면 좋은 정보

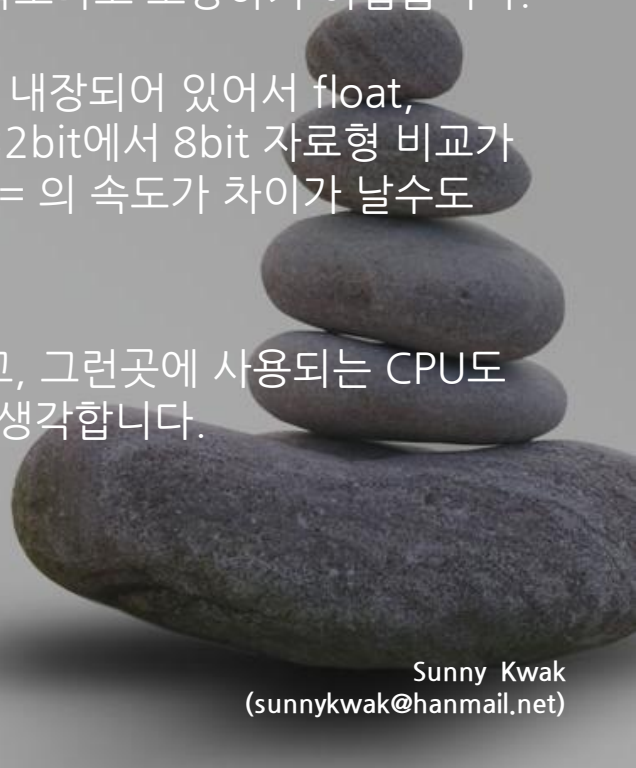
- 거의 의미없는 차이라고 할수는 있지만, int 형이 아닌 경우에는 ==, != 에서 속도 차이가 있을 수 있습니다.

- CPU 아키텍처 (RISC, CISC) 및 레지스터, 어셈블리 내장 명령 셋(set)에 따라 달라질 수 있지만, 예를 들어 32bit 머신에서 64bit비교는 좀 다르죠. 특히 float, double 같은 경우에는 숫자 표현 방식이 다르기 때문에 ==, != 가 동일속도라고 보장하기 어렵습니다.

  물론 인텔 CPU라고 가정하면 요즘에는 모두 co-process가 내장되어 있어서 float, double 연산도 cpu 명령셋에 내장되지만요. 또한 오히려 32bit에서 8bit 자료형 비교가 더 오래걸릴 수 있는 경우도 있습니다. 그런 경우에는 ==, != 의 속도가 차이가 날수도 있습니다.

- 역으로 생각하면 8bit, 16bit embeded 환경도 있는 것이고, 그런곳에 사용되는 CPU도 있기 때문에 절대적이라고 생각하는 것은 좀 조심스럽다고 생각합니다.

  - 금동철님의 조언 -



Sunny Kwak
(sunnykwak@hanmail.net)