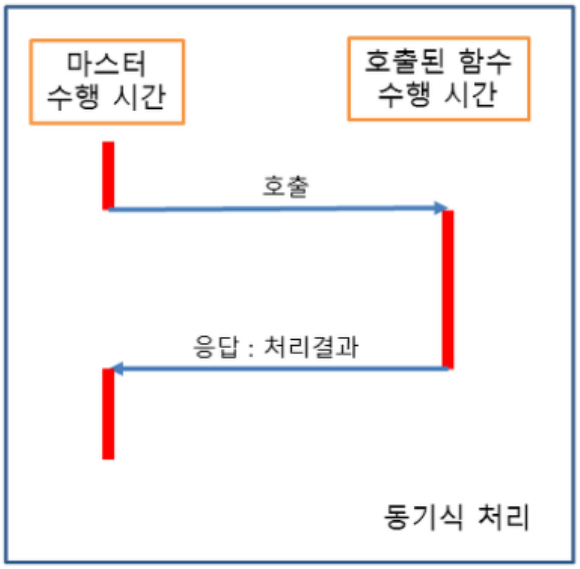
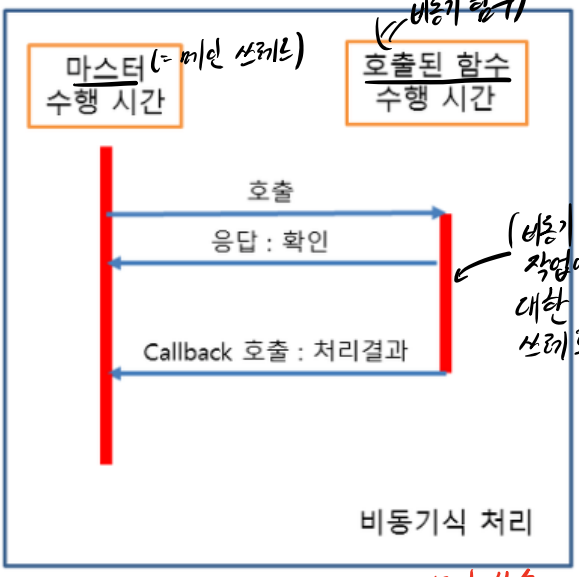
들어가며

JavaScript의 세계에서는 거의 대부분의 작업들이 비동기로 이루어진다. 어떤 작업을 요청하면서 콜백 함수를 등록하면, 작업이 수행되고 나서 결과를 나중에 콜백 함수를 통해 알려주는 식이다. 실제 비동기 작업이 아니더라도 JavaScript의 세계에서는 결과를 콜백으로 알려주는 패턴이 매우 흔하게 사용된다.





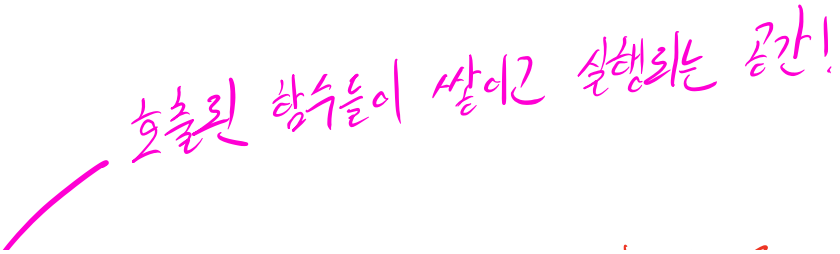
비동기식 처리에 대한 부가적인 설명을 하자면, 마스터가 비동기식 처리를 요청했을 때 호출받은 함수는 바로 응답을 수행한다. 이 응답은 처리 결과에 대한 응답이 아니라 마스터에게 요청을 잘 받았다.라고 확인하는 동작일 뿐이다. 그리고 호출받은 함수는 이미 응답을 했기 때문에 처리결과를 함수 호출이라는 형태로 전달하는 것이다.

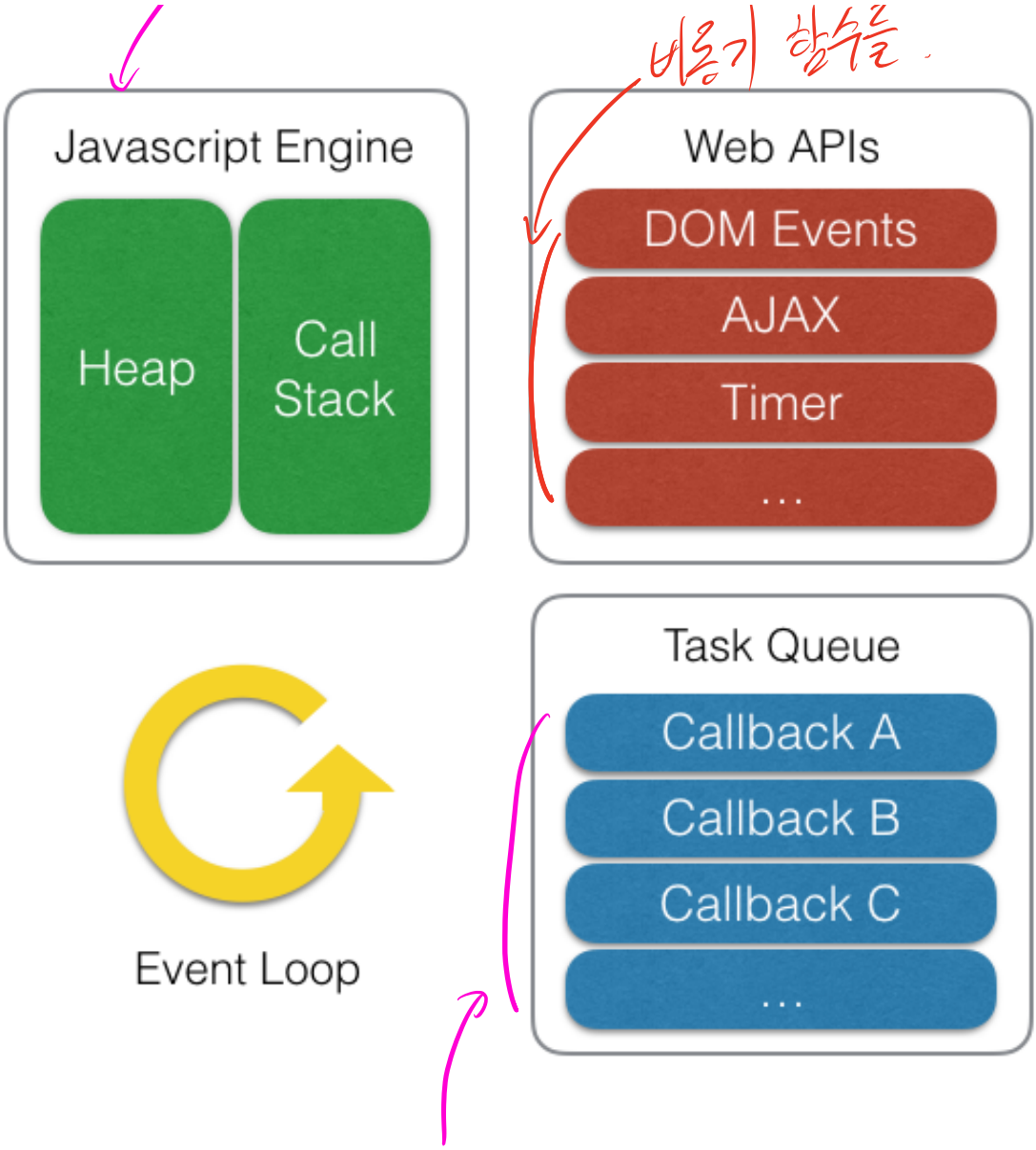
비동기(Asynchronous) 함수란 쉽게 설명하면 호출부에서 실행 결과를 가다리지 않아도 되는 함수입니다.
반대로 동기 함수(Synchronous) 함수는 호출부에서 실행 결과가 리턴될 때 까지 기다려야 하는 함수입니다.

비동기 함수의 이러한 Non-blocking 이점 때문에, 자바스크립트처럼 싱글 쓰레드 환경에서 실행되는 언어에서 광범위하게 사용됩니다.
예를 들어, 브라우저에서 어떤 로직이 동기 함수만으로 실행될 경우, 기다리는 시간이 많아져서 사용자 경험에 부정적인 영향을 미칠 것입니다.
또한, 비동기 함수를 사용하면 로직을 순차적으로 처리할 필요가 없기 때문에 동시 처리에서도 동기 함수 대비 유리한 것으로 알려져있습니다.

하지만 코딩을 하는 개발자 입장에서는 비동기 함수는 동기 함수에 비해서 좀 덜 직관적으로 느껴질 수 있습니다.
동기 함수처럼 순차적 처리가 보장되지 않기 때문에 아래에 위치한 코드가 위에 위치한 코드보다 먼저 실행될 수 있기 때문입니다.

자바 스크립트에는 `setTimeout()` 이라는 대표적인 내장 비동기 함수가 있습니다.
`setTimeout()` 은 두 개의 매개 변수를 받는데, 첫번째는 실행할 작업 내용을 담은 콜백 함수이고, 두번째는 이 콜백 함수를 수행하기 전에 기다리는 밀리초 단위 시간입니다.
즉, `setTimeout()` 함수는 두번째 인자로 들어온 시간 만큼 기다린 후에 첫번째 인자로 들어온 콜백 함수를 실행해줍니다.





비동기 함수들.

백그라운드에서 비동기 작업이 처리된 후에,
해당 비동기 함수의 콜백함수가 `Task Queue'로
전달된다.