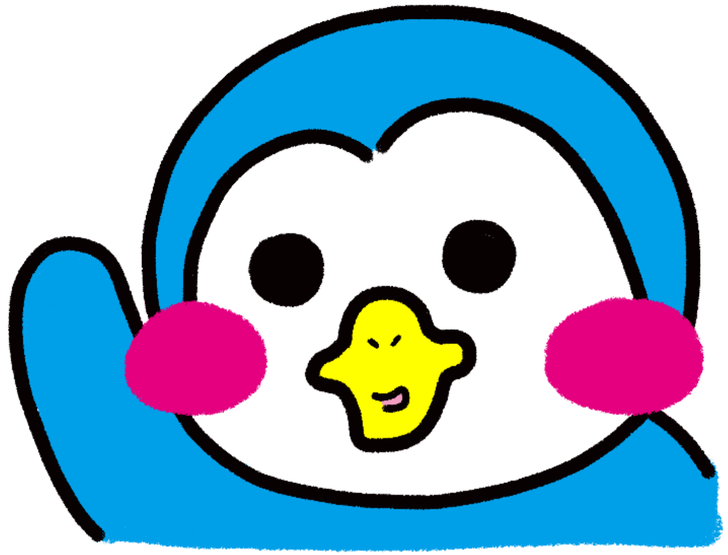
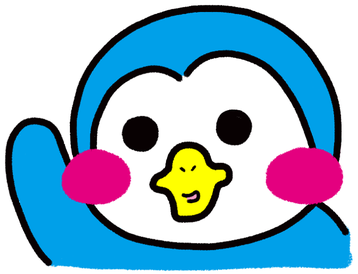
약속은
나중을 위해서



프로미스

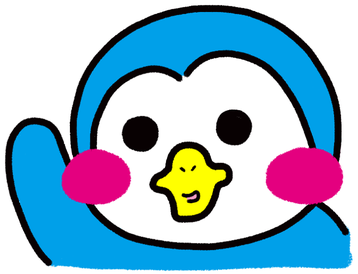
프로미스



promise 객체는 비동기 작업에 대한 약속이다.
비동기 작업 이후 발생할 완료 또는 실패, 그리고 그 결과 값을 나타낸다.

비동기 작업이란?
언제 끝날지 알 수 없는 작업.
작업의 상태가 시작, 진행중, 완료 인 것과 무관하게 프로그램은
계속해서 실행되어야 한다.

프로미스 생성 방법



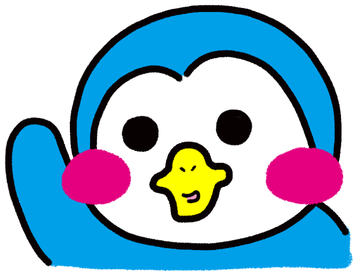
1. 다음과 같이 비동기 작업을 수행하는 함수가 있다.

```
() => {
  console.log("비동기 함수라고 치고")
}
```

2. 이를 Promise 객체(약속)를 생성하면서 인자로 전달한다.

```
new Promise(() => {
  console.log("비동기 함수라고 치고")
})
```

프로미스 생성 방법



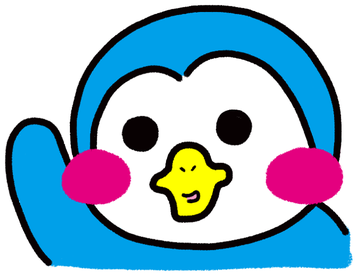
3. Promise 객체에 전달된 함수는 두 가지 콜백 함수를 인자로 받는다.
하나는 작업 성공 시 실행되는 함수(resolve), 하나는 실패 시 함수(reject).

```
new Promise((resolve, reject) => {
  console.log("비동기 함수라고 치고")
})
```

4. 여기에 변수를 선언하여 프로미스 생성 완료!

```
const promise = new Promise((resolve, reject) => {
  console.log("비동기 함수라고 치고")
})
```

성공과 실패

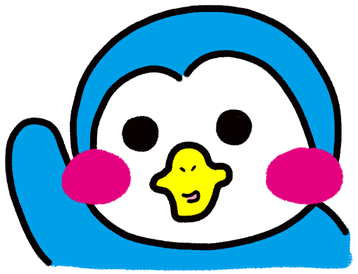


성공 시 콜백 함수와 실패 시 콜백 함수는 비동기 작업으로부터 값을 반환할 수 있다.

```
const isSuccess = true;
const promise = new Promise((resolve, reject) => {
  if(isSuccess){ resolve("성공") }
  else{ reject("실패") }
})
```

=> 위 경우 비동기 작업으로부터 “성공” 이 반환된다.

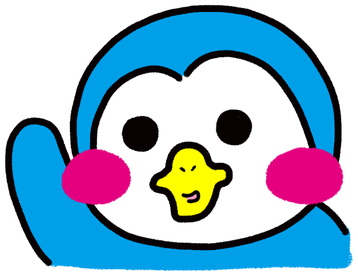
성공과 실패



비동기 작업 완료 후(성공 또는 실패) 이를 처리하기 위해 체이닝(chaning)을 이용하여 해결한다.

```
// then은 성공했을 때, catchsms 실패했을 때
promise.then(function(resolveValue){
  console.log(resolveValue) // 성공
}).catch(function(rejectValue){
  console.log(rejectValue) // 실패
})
```

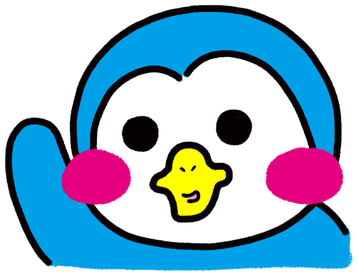
이해를 돕기 위한 예제



```
const promise = new Promise((resolve, reject) => {
  setTimeout(() => {
    console.log("하하")
    resolve("성공")
  }, 2000)
})

promise.then(function(resolveValue){
  console.log("성공 메시지 : ", resolveValue) // 성공
})
console.log("!")
console.log("!")
console.log("!")
console.log("!")
console.log("!")
```

내용 정리



- 프로미스는 비동기 작업에 대한 약속이다.
- 비동기 작업이란 프로그램 내 다른 작업들과 동기가 일치하지 않는 동작을 의미한다.
- 프로미스 객체는 비동기 작업을 전달받아 처리한다.
- 프로미스 객체는 성공 시 콜백함수와 실패 시 콜백함수를 각각 호출할 수 있다.