

프로미스의 3가지 상태(states)

프로미스를 사용할 때 알아야 하는 가장 기본적인 개념이 바로 프로미스의 상태(states)입니다. 여기서 말하는 상태란 프로미스의 처리 과정을 의미합니다. `new Promise()`로 프로미스를 생성하고 종료될 때까지 3가지 상태를 갖습니다.

- **Pending(대기)**: 비동기 처리 로직이 아직 완료되지 않은 상태
- **Fulfilled(이행)**: 비동기 처리가 완료되어 프로미스가 결과 값을 반환해준 상태
- **Rejected(실패)**: 비동기 처리가 실패하거나 오류가 발생한 상태

Pending(대기)

먼저 아래와 같이 `new Promise()` 메서드를 호출하면 Pending(대기) 상태가 됩니다.

```
new Promise();
```

이렇게 `new Promise()` 메서드를 호출할 때 콜백 함수의 인자로 `resolve`, `reject`에 접근할 수 있습니다.

```
new Promise(function (resolve, reject) {  
  // ...  
});
```

Fulfilled(이행)

여기서 콜백 함수의 인자 `resolve`를 아래와 같이 실행하면 Fulfilled(이행) 상태가 됩니다.

```
new Promise(function (resolve, reject) {  
  resolve();  
});
```

그리고 이행 상태가 되면 아래와 같이 `then()`을 이용하여 처리 결과 값을 받을 수 있습니다.

```
function getData() {  
  return new Promise(function (resolve, reject) {  
    var data = 100;  
    resolve(data);  
  });  
}  
  
// resolve()의 결과 값 data를 resolvedData로 받음  
getData().then(function (resolvedData) {  
  console.log(resolvedData); // 100  
});
```

프로미스의 '이행' 상태를 좀 다르게 표현해보면 '완료'입니다.

Rejected(실패)

`new Promise()`로 프로미스 객체를 생성하면 콜백 함수 인자로 `resolve`와 `reject`를 사용할 수 있다고 했습니다. 여기서 `reject` 인자로 `reject()` 메서드를 실행하면 `Rejected(실패)` 상태가 됩니다.

```
new Promise(function (resolve, reject) {  
  reject();  
});
```

그리고, 실패 상태가 되면 실패한 이유(실패 처리의 결과 값)를 `catch()`로 받을 수 있습니다.

```
function getData() {  
  return new Promise(function (resolve, reject) {  
    reject(new Error("Request is failed"));  
  });  
}  
  
// reject()의 결과 값 Error를 err에 받음  
getData().then().catch(function (err) {  
  console.log(err); // Error: Request is failed  
});
```