비동기 프로그래밍

1. 동기(Synchronous)적이란

코드가 반드시 작성된 순서 그대로 실행됨.

비유 -> 동일한 기찻길에 놓인 열차들

2. 비동기(Asynchronous)적이란

어떤 일이 완료되기를 기다리지 않고 다음 코드를 실행해 나가는 프로그래밍 방식

쓰레드나 프로세스가 여럿이 돌고 있음(멀티태스킹 구현)

비유 -> 비동기 방식은 상황에 따라서 이동이 느리거나 자주 서는 열차를 다른 선로에 배치하는 것

3. Callback 함수 란?

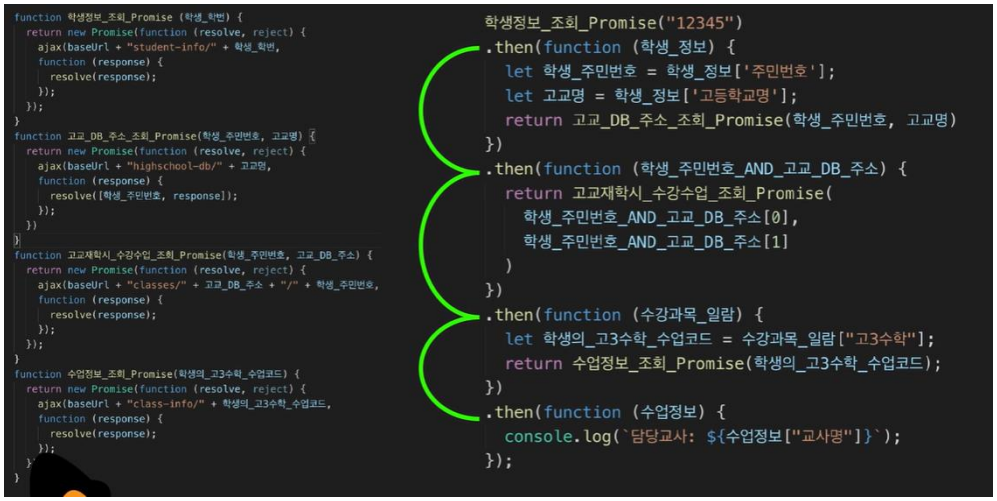
비동기로 주어진 일을 다 마친 다음에 실행되는 함수

4. Web API 란?

타이머를 사용하는 작업, AJAX로 http 요청을 보내거나 파일에서 데이터를 읽어오는 등 시간을 소요하는 작업들을 수행함.

5. Promise

콜백 지옥의 문제를 해결하기 위해 javascript 는 ES6 버전부터 Promise 를 도입함.



```
function 학생정보_조회_Promise (학생_학번) {
  return new Promise(function (resolve, reject) {
    ajax(baseUrl + "student-info/" + 학생_학번,
    function (response) {
      resolve(response);
    });
  });
}
function 고교_DB_주소_조회_Promise(학생_주민번호, 고교명) {
  return new Promise(function (resolve, reject) {
    ajax(baseUrl + "highschool-db/" + 고교명,
    function (response) {
      resolve([학생_주민번호, response]);
    });
  })
}
function 고교재학시_수강수업_조회_Promise(학생_주민번호, 고교_DB_주소) {
  return new Promise(function (resolve, reject) {
    ajax(baseUrl + "classes/" + 고교_DB_주소 + "/" + 학생_주민번호,
    function (response) {
      resolve(response);
    });
  });
}
function 수업정보_조회_Promise(학생의_고3수학_수업코드) {
  return new Promise(function (resolve, reject) {
    ajax(baseUrl + "class-info/" + 학생의_고3수학_수업코드,
    function (response) {
      resolve(response);
    });
  }
}
```

```
학생정보_조회_Promise("12345")
.then(function (학생_정보) {
  let 학생_주민번호 = 학생_정보['주민번호'];
  let 고교명 = 학생_정보['고등학교명'];
  return 고교_DB_주소_조회_Promise(학생_주민번호, 고교명)
})
.then(function (학생_주민번호_AND_고교_DB_주소) {
  return 고교재학시_수강수업_조회_Promise(
    학생_주민번호_AND_고교_DB_주소[0],
    학생_주민번호_AND_고교_DB_주소[1]
  )
})
.then(function (수강과목_일람) {
  let 학생의_고3수학_수업코드 = 수강과목_일람["고3수학"];
  return 수업정보_조회_Promise(학생의_고3수학_수업코드);
})
.then(function (수업정보) {
  console.log(`담당교사: ${수업정보["교사명"]}`);
});
```

Javascript ES7 에서는 Async/Await 이란 기능이 추가됨

Promise로 작성된 코드들을 간결하고 직관적이게 실행 할 수 있음.