**비동기 프로그래밍**

동기 : 하나의 일이 끝나면 다시 일을 처리하는 방식 > 순차적으로 발생하는 것

* 프로그램 소스가 작성된 순서대로 처리를 하는 방식

비동기 : 계속 일이 들어오면서 일을 처리하는 방식

* 프로그램 소스의 작성 순서와 상관없이 처리를 할 수 있는 방식

AJAX : Asynchronous Javascript And Xml

자바스크립트는 원래 태생이 동기일까요 비동기일까요? >>>> 동기이다!

이런 걸 지원할 수 있는 것이 AJAX이다.

싱글"스레드" = "실"

스레드 : 프로그램에서 어떠한 프로세스를 실행하는 단위를 가르킨다

> 하나의 프로세스를 실행한다면 단일 혹은 싱글스레드

> 한번에 여러개의 프로세스를 실행한다면 멀티스레드

자바스크립트는 태생적으로 싱글스레드이다.

동기처리방식 지원 프로그램 언어!

> 콜백함수는 그런 자바스크립트를 보완하기 위해서 멀티스레드를 할 수 있도록 해준다!!!

★ 프로미스 객체 ★ > 내장객체!

Promise : 무분별한 콜백함수 사용의 부작용을 최소화하기 위해서 만든 객체이다

프로미스 객체 => 매개변수 : 조건값이 true일때 실행할 함수 & 조건값이 false일때 실행할 함수

let likePizza = false;

const pizza = new Promise((resolve, reject) => {

  if (likePizza) {

    resolve("피자를 주문합니다.")

  } else {

    reject ("피자를 주문하지 않았습니다.")

  }

});

// pizza는 인스턴스 객체임

pizza

  .then (

    result => console.log(result)

  )

  .catch (

    err => console.log(err)

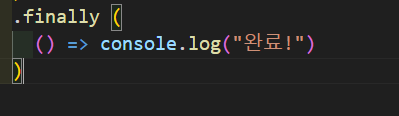
  );

결과



Promise 객체는 then과 catch의 메서드가 꼭 와야 한다.

여기서 인자값 ( result / err )은 다른 단어를 넣어도 됌!



여기서 마지막에 finally() 메서드까지 넣어주면 100점!!!!!

**promise의 객체 진행 단계 (3가지)**

1. pending : promise 객체를 만들고 대기중인 상태

2. fufilled : promise 객체를 활용해서 true값에 도착한 상태

3. rejected : promise 객체를 활용해서 false값에 도착한 상태

// 콘솔창에 피자 준비 띄워보기~~~~ 더 쉽게 쓰는 법!

const pizza = () => {

  return new Promise((resolve, reject) => {

    resolve ("피자를 주문합니다.");

  });

};

const step1 = (message) => {

  console.log(message);

  return new Promise((resolve, reject) => {

    setTimeout(() => {

      resolve("피자 도우 준비");

    }, 3000);

  })

}

const step2 = (message) => {

  console.log(message);

  return new Promise((resolve, reject) => {

    setTimeout(() => {

      resolve("토핑 완료");

    }, 3000);

  })

}

const step3 = (message) => {

  console.log(message);

  return new Promise((resolve, reject) => {

    setTimeout(() => {

      resolve("굽기 완료");

    }, 3000);

  })

}

pizza()

  .then ((result) => step1(result))

  .then ((result) => step2(result))

  .then ((result) => step3(result))

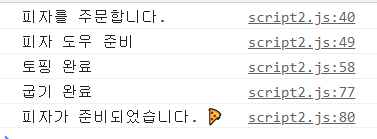
  .then ((result) => console.log(result))

  .then(() => {

    console.log("피자가 준비되었습니다. 🍕")

  })

결과!!!!!!

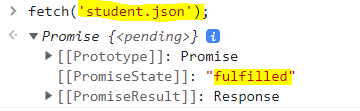


서버에 있는 JSON파일을 가져올 때 사용하는 객체 이름은?

XMLHttpRequest (\* 프라미스 객체 반환이 불가능) = > fetch (\* 프라미스 객체 반환이 가능)

프로미스 객체의 3단계 진행 과정

1. **준비 pending**
2. **완료 fullfiled**
3. **거절 reject**



Fetch 사용해보기 ^^ XMLHttpRequest 얘랑 비교해보기

// 여기서 공부할것 index1 xhr이랑 사용한거 비교하기!

fetch('student.json') // fetch는 오픈/센드 메서드 필요없이 이게 끝..

.then(response => response.json())

.then(json => {

  let output = "";

  json.forEach(student => {

    output += `

    <h2>${student.name}</h2>

    <ul>

    <li>전공 : ${student.major}</li>

    <li>학년 : ${student.grade}학년</li>

    </ul>

    `

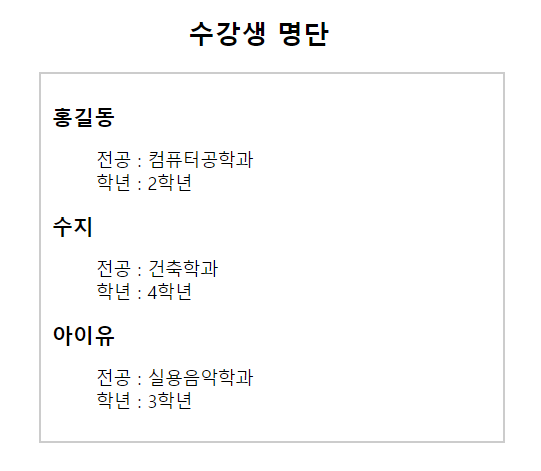
  });

  document.querySelector("#result").innerHTML = output;

})

.catch(err => console.log(err));

결과



**정리는 해보자 ^^**

https 프로토콜

클라이언트 / 서버

GET / POST

XML / JSON

예외처리방식 (try / catch / finally / throw)

비동기 프로그래밍

동기 처리 방식

콜백함수

promise 객체 : 콜백함수의 단점을 보완하기위해 나옴 : 인자값이 2개가 필요함 왜? true일때, false일때! > then() / catch() 이건 내장객체

Fetch (\*\* XMLHttpRequest 업그레이드 버전)

여기서 또 하나가 추가가 되는데..... 그건 바로

**async 함수 / await 예약어**

* async함수와 awaite예약어는 프라미스 객체 혹은 콜백함수 사용 시, 함수 안에 있는 실행문을 비동기적으로 실행하기 위해서 태어났다!

function whatYourFavorit() {

  let fav = "javascript"

  return new Promise((resolve, reject) => resolve(fav));

}

function displaySubject(yyy) {

  return new Promise((resolve, reject) => resolve(`Hello, ${yyy}`));

}

whatYourFavorit()

.then(displaySubject)

.then(console.log)

결과



Async로 완전 간단하게 출력하기!!!!

// 위에 예제문제를 async 함수 활용해보기

async function whatYourFavorit() {

  let fav = "javascript"

  return fav;

}

async function displaySubject(yyy) {

  return `Hello, ${yyy}`;

}

whatYourFavorit()

.then(displaySubject)

.then(console.log)

결과,,,,,,,,,,,,,,,,,,,는 똑같음 !



Await 사용해보기

async function whatYourFavorit() {

  let fav = "javascript"

  return fav;

}

async function displaySubject(yyy) {

  return `Hello, ${yyy}`;

}

async function init() {

  const response = await whatYourFavorit();

  const result = displaySubject(response)

  console.log(result)

}

init();

await 예약어.,.쓰는 이유

whatYourFavorit()함수의 실행이 끝날때까지 기다린 후 response값을 저장

response 값을 사용해서 displaySubject()함수를 실행하고 결과값을 result에 저장하기 위해 사용하는 것이다.

async 함수는 기본적으로 비동기처리방식으로 움직이므로

동기처리방식으로 하기 위해서 순차적으로 데이터를 처리해야할 땐

await을 쓰는 것이다!!!!!!!!!!

async 함수를 쓰지 않으면 await을 지정하지 않아도 된다