**async/await**

* ES8에 해당하는 문법으로서, promise를 더욱 쉽게 사용할 수 있음
* Promise 가 끝날때까지 await가 기다렸다가 다음 작업을 수행
* 사용법

//Promise를 반환하는 함수 앞에 await를 사용한다.

//await를 사용하는 함수는 앞에 async를 붙인다.

function sleep(ms) {

return new Promise(resolve => setTimeout(resolve,ms));

}

async function process() {

console.log(“안녕하세요”);

await sleep(1000);

console.log(“반갑습니다”);

}

* async함수는 결과값으로 Promise를 반환함. 따라서 다음 코드도 가능

return 값이 resolve 값 / throw 문이 reject 값으로 생각하고, 각각 then, catch로 잡아낸다.

process().then(() => {

console.log(“done!”);

});

* async함수안의 await에서 발생하는 에러를 잡아낼 때는 try/catch문으로 잡는다.

async function makeError() {

await sleep(1000);

const error = new Error();

throw error;

}

async function process() {

try {

await makeError();

} catch (e) {

console.error(e);

}

}

* Promise.all : 여러 개의 Promise 작업을 동시에 시작하고 싶을 때 사용한다.

이때 등록한 Promise 중 하나라도 실패하면, 모든게 실패한 것으로 간주됨.

async function process() {

const results = await Promise.all([getDog(), getRabbit(), getTurtle()]);

console.log(results);

}

* Promise.race : 여러 개의 Promise를 등록하고, 가장 빨리 끝난거 하나만의 결과값을 가져옴