비동기 처리

유튜브 주소 : https://youtu.be/iwDgIFL6Hpw

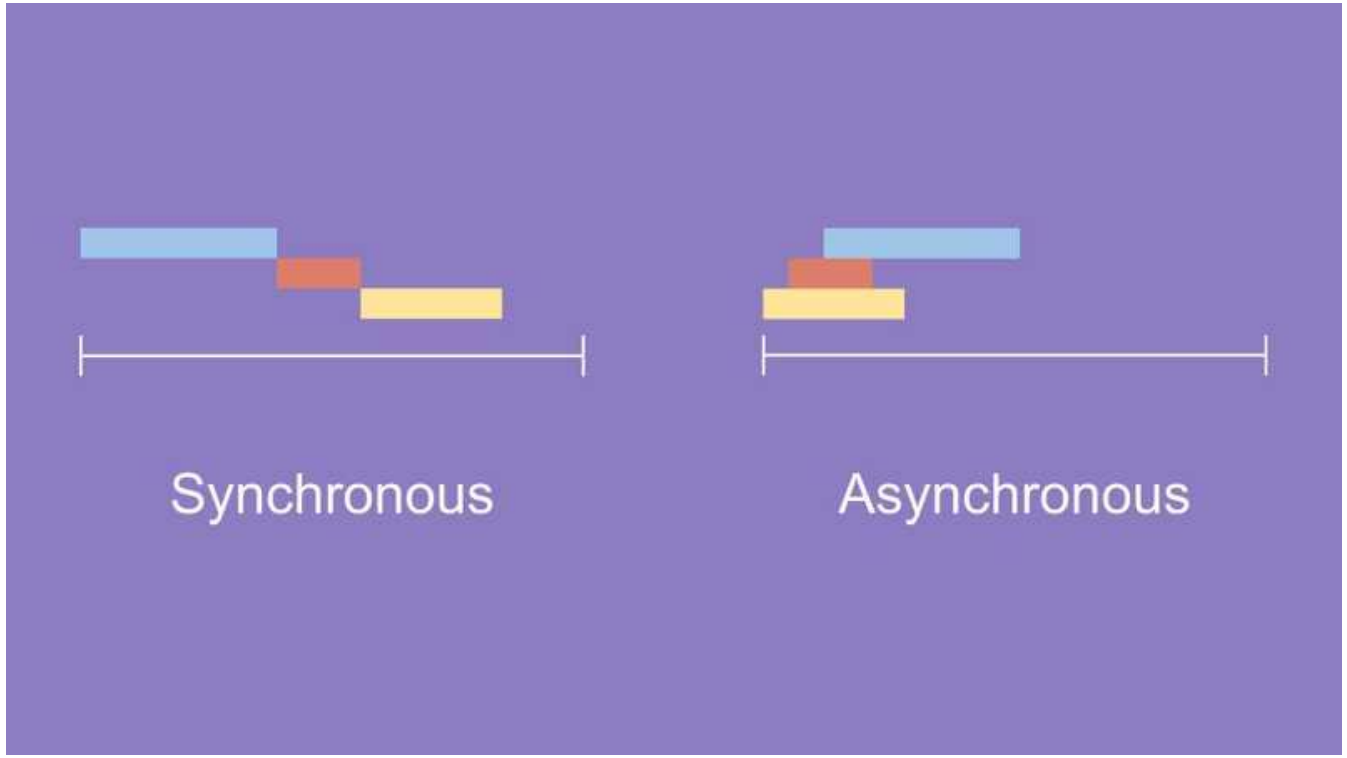
동기와 비동기

비동기 제어(콜백함수)

비동기 제어(Promise + async/await)

동기와 비동기

- 동기
  - 요청을 보낸 후 해당 요청에 대한 응답을 받아야 다음 동작을 실행하는 것
- 비동기
  - 요청을 보낸 후 해당 응답 여부와 관계없이 다음 동작을 바로 실행하는 것



동기와 비동기

- 동기적 처리

```
> console.log("1st");
  console.log("2st");
  console.log("3st");
```

```
1st                                   VM28:1
2st                                   VM28:2
3st                                   VM28:3
```

- 비동기적 처리

```
> console.log("1st");
  setTimeout(()=> {
      console.log("2st");
  },3000)
  console.log("3st");
```

```
1st                                   VM32:1
3st                                   VM32:5
<- undefined
2st                                   VM32:3
```

```
> console.log("1st");
  setTimeout(()=> {
      console.log("2st");
  },0)
  console.log("3st");
```

```
1st                                   VM44:1
3st                                   VM44:5
<- undefined
2st                                   VM44:3
```

동기와 비동기

- 비동기적 처리가 수행되는 경우
  - Ajax Web API 요청
    - 서버 측에서 데이터를 받아와야 하는 경우
  - 파일 읽기
    - 서버에서 파일을 읽어야 하는 경우
  - 암호화 / 복호화
    - 암호화 및 복호화 작업이 바로 처리 되지 않고 시간이 소요되는 경우
  - 작업 예약
    - setTimeout 메소드를 이용하여 비동기 처리하는 경우

콜 백 함수

- CallBack 함수
  - 이벤트 발생, 특정 시점 도달 시 시스템에서 호출하는 함수
  - 코드를 통해 명시적으로 호출하는 함수가 아님(익명 함수)
  - 개발자는 단지 함수를 등록하기만 함
  - **함수 안에서 실행하는 또 다른 함수**

콜 백 함수

- 콜 백 함수만을 바꿔줌으로써 하나의 함수를 여러 가지로 응용 가능

```
function introduce (lastName, firstName, callback) {
let fullName = lastName + firstName;
callback(fullName);
}
function say_hello (name) {
console.log("안녕하세요 제 이름은 " + name + "입니다");
}
function say_bye (name) {
console.log("지금까지 " + name + "이었습니다. 안녕히계세요");
}
introduce("홍", "길동", say_hello);

introduce("홍", "길동", say_bye);
```

콜 백 함수

- Ajax 통신 코드

```
function getData() {
    var tableData;
    $.get('https://hansung.ac.kr/', function (response) {
        tableData = response;
    });
    return tableData;
}

console.log(getData());
```

- 콘솔엔 undefined 출력
  - $.get 메소드가 비동기적 메소드이기 때문에 발생하는 문제

콜 백 함수

- 콜 백 함수를 이용하여 수정한 ajax 통신 코드

```
function getData(callbackFunc) {
    $.get('https://hansung.ac.kr/', function (response) {
        callbackFunc(response);
    });
}

getData(function (tableData) {
    console.log(tableData); //
});
```

  - getData 함수 정의 시 파라미터로 콜 백 함수를 제공
  - 해당 콜백 함수는 $.get 메소드 안에서 실행
  - $.get 메소드는 서버에서 받은 데이터를 콜 백 함수의 인자로 넘겨줌
  - 필요한 데이터가 준비된 시점에서 동작을 수행할 수 있게 제어 가능

콜 백 함수

- 콜 백 지옥?

- 콜 백 함수를 익명함수로 전달되는 과정이 반복되어 코드의 들여쓰기 수준이 감당하기 힘들 정도로 깊어지는 현상

- 어떤 기능의 실행 결과를 받아 또 다른 어떤 기능을 실행해야 하는 상황에 만들어지는 코드

- 가독성 저하, 유지보수 어려움

```
> setTimeout(
    (name) => {
      let coffeeList = name;
      console.log(coffeeList);

      setTimeout(
        (name) => {
          coffeeList += ', ' + name;
          console.log(coffeeList);

          setTimeout(
            (name) => {
              coffeeList += ', ' + name;
              console.log(coffeeList);

              setTimeout(
                (name) => {
                  coffeeList += ', ' + name;
                  console.log(coffeeList);
                },
                500,
                'Latte',
              );
            },
            500,
            'Mocha',
          );
        },
        500,
        'Americano',
      );
    },
    500,
    'Espresso',
  );
< 210
  Espresso                                  instrument.ts:113
  Espresso, Americano                       instrument.ts:113
  Espresso, Americano, Mocha                instrument.ts:113
  Espresso, Americano, Mocha, Latte         instrument.ts:113
```

비동기 제어(Promise)

- 비동기 작업의 결과(성공 / 실패)를 나타내는 객체

- JS ES2015 버전부터 등장
  - JS와 노드의 API들이 콜 백 대신 Promise 기반으로 재구성

- 프로미스 객체를 생성하여 사용
  - 프로미스 객체는 3가지 상태를 지님
    - Pending – 초기상태(대기)
    - Fulfilled – 성공한 상태(.then으로)
    - Rejected – 실패한 상태(.catch로)

    - .then -> 성공(resolve)한 경우 실행
    - .catch -> 실패(reject)한 경우 실행

비동기 제어(Promise)

- Promise를 사용해 수정한 코드

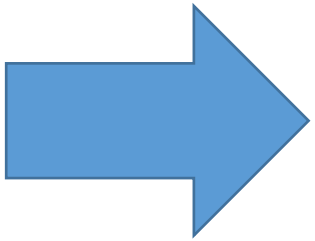
콜 백 함수

```
> setTimeout(
    (name) => {
      let coffeeList = name;
      console.log(coffeeList);

      setTimeout(
        (name) => {
          coffeeList += ', ' + name;
          console.log(coffeeList);

          setTimeout(
            (name) => {
              coffeeList += ', ' + name;
              console.log(coffeeList);

              setTimeout(
                (name) => {
                  coffeeList += ', ' + name;
                  console.log(coffeeList);
                },
                500,
                'Latte',
              );
            },
            500,
            'Mocha',
          );
        },
        500,
        'Americano',
      );
    },
    500,
    'Espresso',
  );
< 210
Espresso                                    instrument.ts:113
Espresso, Americano                         instrument.ts:113
Espresso, Americano, Mocha                  instrument.ts:113
Espresso, Americano, Mocha, Latte           instrument.ts:113
```



Promise 사용

```
> new Promise((resolve) => {
    setTimeout(() => {
      let name = 'Espresso';
      console.log(name);
      resolve(name);
    }, 500);
  })
    .then((prevName) => {
      return new Promise((resolve) => {
        setTimeout(() => {
          let name = prevName + ', Americano';
          console.log(name);
          resolve(name);
        }, 500);
      });
    })
    .then((prevName) => {
      return new Promise((resolve) => {
        setTimeout(() => {
          let name = prevName + ', Mocha';
          console.log(name);
          resolve(name);
        }, 500);
      });
    })
    .then((prevName) => {
      return new Promise((resolve) => {
        setTimeout(() => {
          let name = prevName + ', Latte';
          console.log(name);
          resolve(name);
        }, 500);
      });
    });
< ▶ Promise {<pending>}
Espresso                                    instrument.ts:113
Espresso, Americano                         instrument.ts:113
Espresso, Americano, Mocha                  instrument.ts:113
Espresso, Americano, Mocha, Latte           instrument.ts:113
```

비동기 제어(Promise + async/await)

- Promise -> then 지옥 발생 가능?
- Promise -> 콜백의 단점 해소, 그러나 then, catch가 반복됨
- 기존 콜 백과 Promise의 단점을 해소하고자 만들어짐
- ES2017 버전에서 추가

비동기 제어(Promise + async/await)

- Async
  - function() 앞에 async 키워드 추가하여 사용
  - Await 키워드가 비동기 코드를 호출할 수 있게 해주는 함수
  - Async 함수를 실행하게 되면 무조건 Promise 객체가 반환됨
  - Async 함수 내에서 return값은 반환된 Promise 객체의 결과(resolve)값

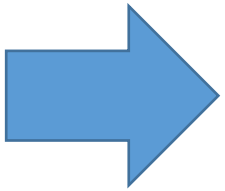
- Await
  - 반드시 async 함수 내에서만 사용 가능
  - 일반 함수에서 사용하면 문법 오류 발생
  - await 키워드는 Promise 객체를 생성하는 함수 앞에 사용 가능
  - JS가 await 키워드를 만나게 되면 해당 함수가 Promise 상태가 이행될 때까지 기다린 후 이행이 완료되면 결과값을 반환 한 후 다음 코드 실행

비동기 제어(Promise + async/await)

- async/await를 이용한 코드

Promise

```
> new Promise((resolve) => {
    setTimeout(() => {
      let name = 'Espresso';
      console.log(name);
      resolve(name);
    }, 500);
  })
    .then((prevName) => {
      return new Promise((resolve) => {
        setTimeout(() => {
          let name = prevName + ', Americano';
          console.log(name);
          resolve(name);
        }, 500);
      });
    })
    .then((prevName) => {
      return new Promise((resolve) => {
        setTimeout(() => {
          let name = prevName + ', Mocha';
          console.log(name);
          resolve(name);
        }, 500);
      });
    })
    .then((prevName) => {
      return new Promise((resolve) => {
        setTimeout(() => {
          let name = prevName + ', Latte';
          console.log(name);
          resolve(name);
        }, 500);
      });
    });
◂ ▸ Promise {<pending>}
Espresso                                    instrument.ts:113
Espresso, Americano                         instrument.ts:113
Espresso, Americano, Mocha                  instrument.ts:113
Espresso, Americano, Mocha, Latte           instrument.ts:113
```



async/await

```
> const addCoffee = (name) => {
    return new Promise((resolve) => {
      setTimeout(() => {
        resolve(name);
      }, 500);
    });
  };

  const coffeeMaker = async () => {
    let coffeeList = '';
    let _addCoffee = async (name) => {
      coffeeList += (coffeeList ? ', ' : '') + (await
  addCoffee(name));
    };
    await _addCoffee('Espresso');
    console.log(coffeeList);
    await _addCoffee('Americano');
    console.log(coffeeList);
    await _addCoffee('Mocha');
    console.log(coffeeList);
    await _addCoffee('Latte');
    console.log(coffeeList);
  };

  coffeeMaker();
◂ ▸ Promise {<pending>}
Espresso                                    instrument.ts:113
Espresso, Americano                         instrument.ts:113
Espresso, Americano, Mocha                  instrument.ts:113
Espresso, Americano, Mocha, Latte           instrument.ts:113
```

Q & A