02. 비동기와 API

async와 await

async와 await

await

```
const start = async () => {
    delay(3000).then((res) => {
        console.log(res);
    });
};

start();
```

```
const start = async () => {
    let result = await delay(3000);
    console.log(result);
};

start();
```

async와 await

await

```
const workA = () => {
    setTimeout(() => {
        console.log('workA');
    }, 5000);
};
const workB = () => {
    setTimeout(() => {
        console.log('workB');
    }, 3000);
};
const workC = () => {
    setTimeout(() => {
        console.log('workC');
    }, 10000);
};

workA();
workB();
workC();
```

```
workB    index.js:8
workA    index.js:3
workC    index.js:13
```

```
const workA = () => {
    return new Promise((resolve) => {
        setTimeout(() => {
            resolve('workA');
        }, 5000);
    });
};
const workB = () => {
    return new Promise((resolve) => {
        setTimeout(() => {
            resolve('workB');
        }, 3000);
    });
};
const workC = () => {
    return new Promise((resolve) => {
        setTimeout(() => {
            resolve('workC');
        }, 10000);
    });
};

const start = async () => {
    try {
        let resultA = await workA();
        let resultB = await workB();
        let resultC = await workC();
        console.log(resultA);
        console.log(resultB);
        console.log(resultC);
    } catch (err) {
        console.log(err);
    }
};

start();
```

```
workA index.js:28
workB index.js:29
workC index.js:30
```

async와 await

await

```
const start = async () => {
    try {
        let resultA = await workA();
        let resultB = await workB();
        let resultC = await workC();
        console.log(resultA);
        console.log(resultB);
        console.log(resultC);
    } catch (err) {
        console.log(err);
    }
};

start();
```

```
const start = () => {
    try {
        let resultA = await workA();
        let resultB = await workB();
        let resultC = await workC();
        console.log(resultA);
        console.log(resultB);
        console.log(resultC);
    } catch (err) {
        console.log(err);
    }
};

start();
```

```
Uncaught                                    index.js:25
SyntaxError: await is only valid in async
functions and the top level bodies of
modules (at index.js:25:23)
```

async와 await

await

```
const start = async () => {
    try {
        let resultA = await workA();
        let resultB = await workB();
        let resultC = await workC();
        console.log(resultA);
        console.log(resultB);
        console.log(resultC);
    } catch (err) {
        console.log(err);
    }
};

start();
```

```
const start = () => {
    try {
        let resultA = await workA();
        let resultB = await workB();
        let resultC = await workC();
        console.log(resultA);
        console.log(resultB);
        console.log(resultC);
    } catch (err) {
        console.log(err);
    }
};

start();
```

```
Uncaught                                    index.js:25
SyntaxError: await is only valid in async
functions and the top level bodies of
modules (at index.js:25:23)
```