Promise와 비동기 완전 정복

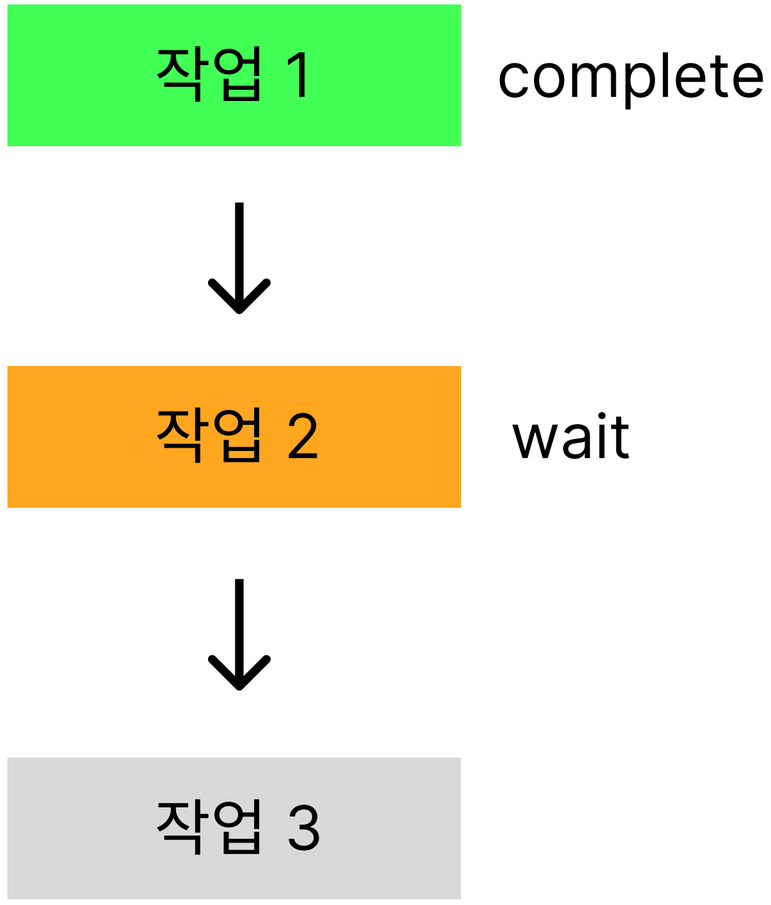
동기 vs 비동기

콜백

프로미스

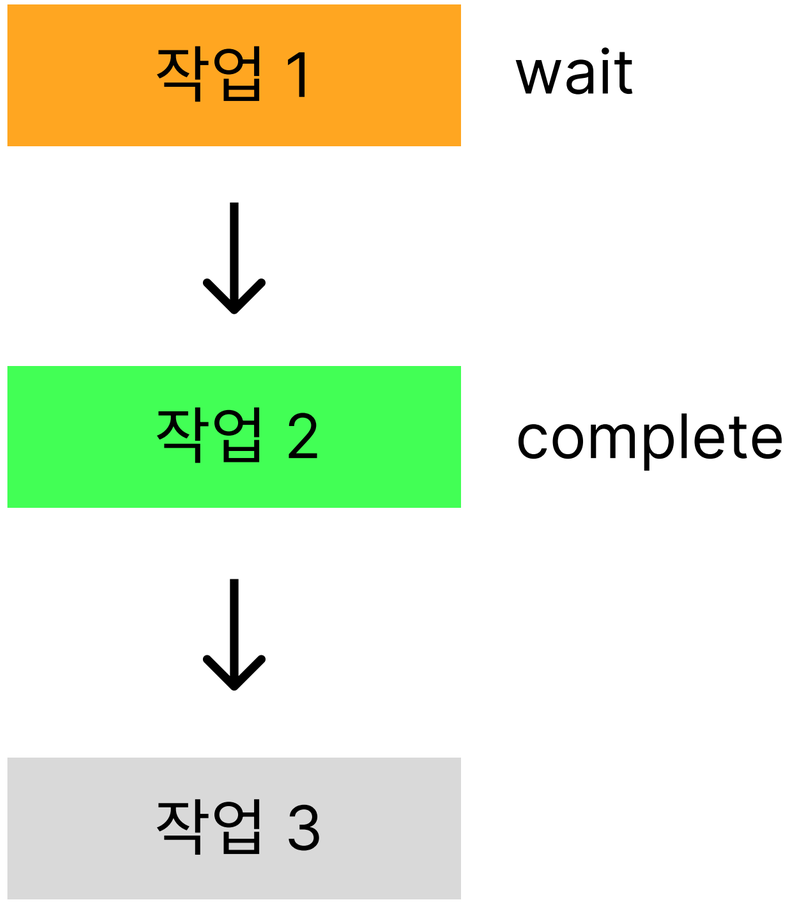
동기적 수행

- 한번에 하나씩
- 순서대로



비동기적 수행

- 특정 기준을 바탕으로 쭉 수행



그래서 무슨 기준인데?

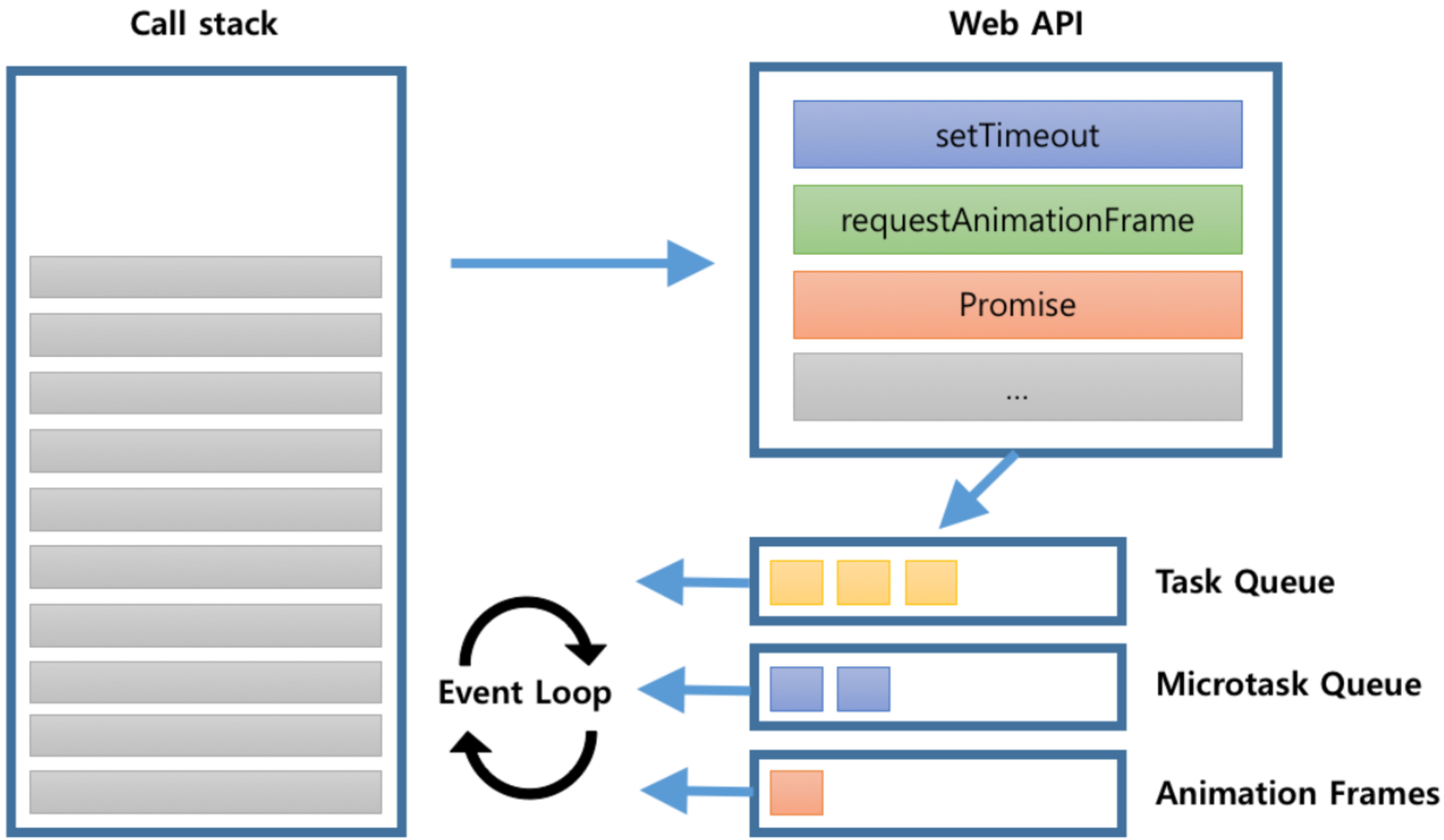
```
console.log("1");

setTimeout(() => {
    console.log("2");
}, 1000);

Promise.resolve(1).then(() => console.log("3"));

console.log("4");
```

이벤트 루프를 알아야 한다



이벤트 루프를 왜 알아야 할까

index.js > ...

```
1  function getData() {
2    setTimeout(() => {
3      console.log('서버에서 데이터를 받아왔어요');
4    }, 2000);
5  }
6
7  getData();
8  console.log('후처리..');
9
```

콜백 함수를 쓰는 이유

index.js > ...

```
function getData(callback) {
  setTimeout(() => {
    console.log('서버에서 데이터를 받아왔어요');
    callback();
  }, 2000);
}

getData(() => {
  console.log('후처리..');
});
```

보아라, 내가 만든 콜백 지옥

```
first(() => {
    second(() => {
        third(() => {
            fourth(() => {
                console.log("done?");
            })
        })
    })
})
```

프로미스는 신이고 무적이다

```
18    first()
19    .then(() => second())
20    .then(() => third())
21    .then(() => fourth())
22    .catch(() => console.error("promise error"));
23
```

비동기 작업의 3가지 상태

대기 성공 실패

프로미스의 3가지 상태

Pending	Fulfilled	Reject
대기	성공	실패

앞에 썼던 resolve는 뭐에요?

JS index.js > promise > <function> > setTimeout() callback

```
1   const promise = new Promise((resolve, reject) => {
2     setTimeout(() => {
3       // const data = { name: '철수' };
4       const data = null;
5       if (data) {
6         console.log('네트워크 요청 성공');
7         resolve(data);
8       } else {
9         reject(new Error('네트워크 문제!!!'));
10      }
11    }, 1000);
12  });
13
14  setTimeout(() => {
15    console.log(promise);
16  }, 2000);
17
```

resolve : 성공 상태 반환

reject : 실패 상태 반환

직접 비동기 함수를 만들어보자

index.js > ...

```
function getData() {
  const promise = new Promise((resolve, reject) => {
    setTimeout(() => {
      const data = { name: '철수' };
      //   const data = null;
      if (data) {
        console.log('네트워크 요청 성공');
        resolve(data);
      } else {
        reject(new Error('네트워크 문제!!!'));
      }
    }, 1000);
  });

  return promise;
}
```

Async === Promise ?

```index.js
// Async

async function getUser() {
  return '별코딩';
}

const user = getUser();
user.then((name) => console.log(name));
```

정상적으로 실행될까?

Async === Promise ?

```index.js
// Async

function networkRequest() {
  return new Promise((resolve) => {
    setTimeout(() => {
      console.log('데이터를 받아왔습니다');
      resolve('서버 1');
    }, 2000);
  });
}

async function getUser() {
  await networkRequest();
  await networkRequest();
  return '별코딩';
}

const user = getUser();
user.then((name) => console.log(name));
```

User.then 코드가 정상적으로 실행될까?

Async === Promise ?

```
 9  async function getUser() {
10    await networkRequest();
11    return '별코딩';
12  }
13
14  async function getTodo() {
15    await networkRequest();
16    return ['청소하기', '밥먹기'];
17  }
18
19  async function getData() {
20    const user = await getUser();
21    const todo = await getTodo();
22    console.log(`${user}님 ${todo}를 하세요`);
23  }
```

정상적으로 실행될까?

감사합니다.

참고자료 : 별코딩 유튜브 - 비동기 프로그래밍