Google Developer Student Clubs

JavaScript #2

Core JavaScript

JINJAE KIM
Jinjae.dev@gmail.com

Promise & async, await

Asynchronous

When?

- Loading Module or Script
- Network Request & Response
- User Event
- File System
- ...

```
console.log("Test started!");
function loadScript(src) {
  let script = document.createElement('script');
  script.src = src;
  document.head.append(script);
}
loadScript("long-time.js");
console.log("Test done!");
```

Asynchronous

Why?

Efficiency !

```
console.log("Test started!");

function loadScript(src, callback) {
  let script = document.createElement('script');
  script.src = src;

  script.onload = () => callback(src);

  document.head.append(script);
}

loadScript("long-time.js", function() {
  console.log("Test done!");
});
```

Error Handling

```
function loadScript(src, callback) {
  let script = document.createElement('script');
  script.src = src;

  script.onload = () => callback(null, script);
  script.onerror = () => callback(new Error(`${src}를 불러오는 도중에 에러가 발생했습니다.`));

  document.head.append(script);
}
```

Error Handling

```
loadScript('/my/script.js', function(error, script) {
  if (error) {
    // 에러 처리
  } else {
    // 스크립트 로딩이 성공적으로 끝남
  }
});
```

Is this a good code?

```
loadScript('1.js', function(error, script) {

  if (error) {
    handleError(error);
  } else {
    // ...
    loadScript('2.js', function(error, script) {
      if (error) {
        handleError(error);
      } else {
        // ...
        loadScript('3.js', function(error, script) {
          if (error) {
            handleError(error);
          } else {
            // 모든 스크립트가 로딩된 후, 실행 흐름이 이어집니다. (*)
          }
        });

      }
    })
  }
});
```

Google Developer Student Clubs

Is this a good code?

```
loadScript('1.js', step1);

function step1(error, script) {
  if (error) {
    handleError(error);
  } else {
    // ...
    loadScript('2.js', step2);
  }
}

function step2(error, script) {
  if (error) {
    handleError(error);
  } else {
    // ...
    loadScript('3.js', step3);
  }
}

function step3(error, script) {
  if (error) {
    handleError(error);
  } else {
    // 모든 스크립트가 로딩되면 다른 동작을 수행합니다. (*)
  }
};
```

Catch!

Error Handling

```
let user = "jin-jae";

fetch(`https://api.github.com/users/${user}`)
  .then(response => response.json())
  .then(githubUser => new Promise((resolve, reject) => {
    let img = document.createElement('img');
    img.src = githubUser.avatar_url;
    img.className = "promise-avatar-example";
    document.body.append(img);

    setTimeout(() => {
      img.remove();
      resolve(githubUser);
    }, 3000);
  }))
  .catch(error => alert(error.message));
```

If there is no catch...

```
new Promise(function() {
  noSuchFunction(); // 존재하지 않는 함수를 호출하기 때문에 에러가 발생함
})
  .then(() => {
    // 성공상태의 프라미스를 처리하는 핸들러. 한 개 혹은 여러 개가 있을 수 있음
  });
```

Use unhandledrejection

```
window.addEventListener('unhandledrejection', function(event) {
  // unhandledrejection 이벤트엔 두 개의 특수 프로퍼티가 있습니다.
  alert(event.promise); // [object Promise] - 에러를 생성하는 프라미스
  alert(event.reason); // Error: 에러 발생! - 처리하지 못한 에러 객체
});

new Promise(function() {
  throw new Error("에러 발생!");
}); // 에러를 처리할 수 있는 .catch 핸들러가 없음
```

Google Developer Student Clubs

MicroTask Queue

What is it?

Asynchronous Task

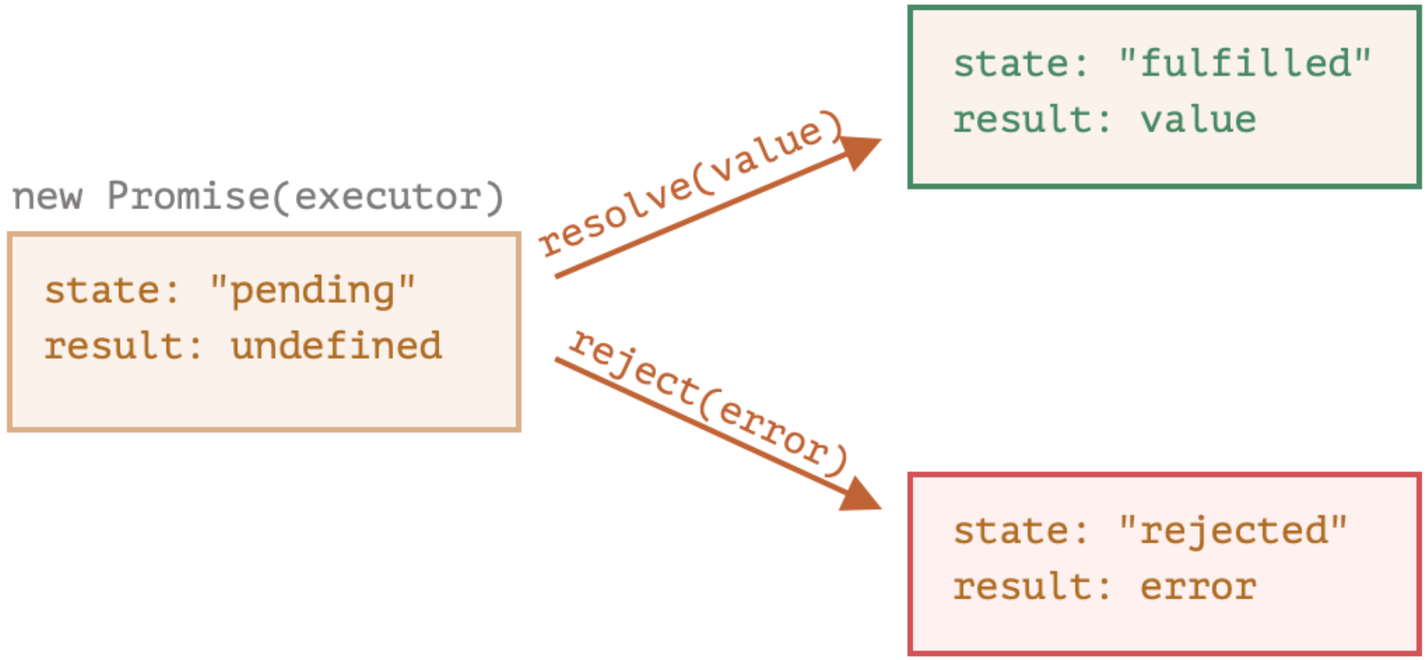
```
let promise = Promise.resolve();

promise.then(() => alert("Success!"));

alert("EOF");
```

Promise

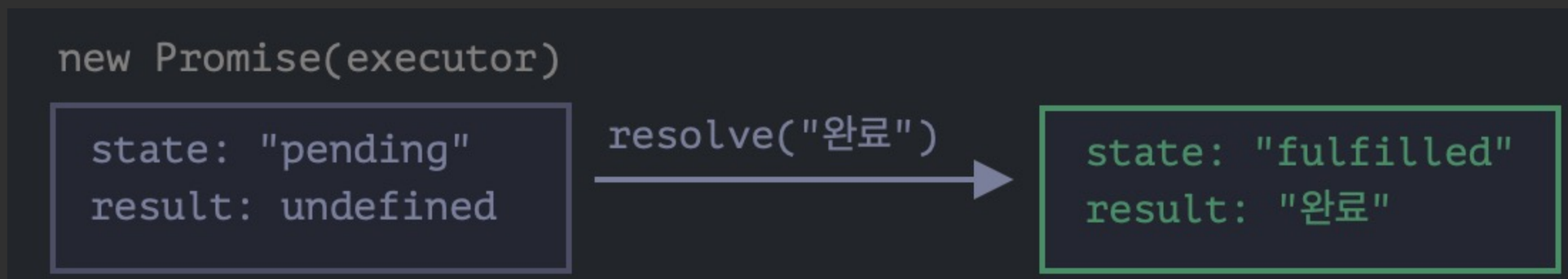
Why?

- Producing code
- Consuming code
- Promise

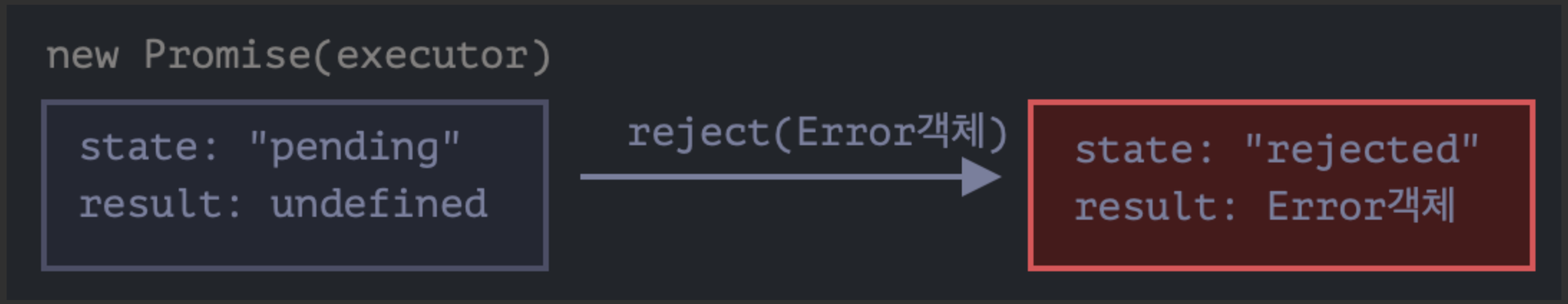
Promise State & Result



```
let promise = new Promise(function(resolve, reject) {
  // 프라미스가 만들어지면 executor 함수는 자동으로 실행됩니다.

  // 1초 뒤에 일이 성공적으로 끝났다는 신호가 전달되면서 result는 '완료'가 됩니다.
  setTimeout(() => resolve("완료"), 1000);
});
```



```
let promise = new Promise(function(resolve, reject) {
  // 1초 뒤에 에러와 함께 실행이 종료되었다는 신호를 보냅니다.
  setTimeout(() => reject(new Error("에러 발생!")), 1000);
});
```



then

```
let promise = new Promise(function(resolve, reject) {
  setTimeout(() => resolve("완료!"), 1000);
});

// resolve 함수는 .then의 첫 번째 함수(인수)를 실행합니다.
promise.then(
  result => alert(result), // 1초 후 "완료!"를 출력
  error => alert(error) // 실행되지 않음
);
```

then

```
let promise = new Promise(function(resolve, reject) {
  setTimeout(() => reject(new Error("에러 발생!")), 1000);
});

// reject 함수는 .then의 두 번째 함수를 실행합니다.
promise.then(
  result => alert(result), // 실행되지 않음
  error => alert(error) // 1초 후 "Error: 에러 발생!"을 출력
);
```

finally

```
et promise = new Promise((resolve, reject) => {
  setTimeout(() => reject(new Error("에러 발생!")), 1000);
});

// .catch(f)는 promise.then(null, f)과 동일하게 작동합니다
promise.catch(alert); // 1초 뒤 "Error: 에러 발생!" 출력
```

Promise – Method

Why?

- Promise.all
- Promise.allSettled
- Promise.race
- Promise.resolve / Promise.result

Promise.all

```
Promise.all([
  new Promise(resolve => setTimeout(() => resolve(1), 3000)), // 1
  new Promise(resolve => setTimeout(() => resolve(2), 2000)), // 2
  new Promise(resolve => setTimeout(() => resolve(3), 1000))  // 3
]).then(alert); // 프라미스 전체가 처리되면 1, 2, 3이 반환됩니다. 각 프라미스는 배열을 구성하는 요소가 됩니다.
```

Promise.allSettled

```
let urls = [
  'https://api.github.com/users/iliakan',
  'https://api.github.com/users/Violet-Bora-Lee',
  'https://no-such-url'
];

Promise.allSettled(urls.map(url => fetch(url)))
  .then(results => { // (*)
    results.forEach((result, num) => {
      if (result.status == "fulfilled") {
        alert(`${urls[num]}: ${result.value.status}`);
      }
      if (result.status == "rejected") {
        alert(`${urls[num]}: ${result.reason}`);
      }
    });
  });
```

Promise.race

```
Promise.race([
  new Promise((resolve, reject) => setTimeout(() => resolve(1), 1000)),
  new Promise((resolve, reject) => setTimeout(() => reject(new Error("에러 발생!")), 2000)),
  new Promise((resolve, reject) => setTimeout(() => resolve(3), 3000))
]).then(alert); // 1
```

async & await

Use Promise more comfortable

async

```
async function f() {
  return 1;
}

f().then(alert); // 1
```

```
async function f() {
  return Promise.resolve(1);
}

f().then(alert); // 1
```

await

```
async function f() {

  let promise = new Promise((resolve, reject) => {
    setTimeout(() => resolve("완료!"), 1000)
  });

  let result = await promise; // 프라미스가 이행될 때까지 기다림 (*)

  alert(result); // "완료!"
}

f();
```

Thenable

```
class Thenable {
  constructor(num) {
    this.num = num;
  }
  then(resolve, reject) {
    alert(resolve);
    // 1000밀리초 후에 이행됨(result는 this.num*2)
    setTimeout(() => resolve(this.num * 2), 1000); // (*)
  }
};

async function f() {
  // 1초 후, 변수 result는 2가 됨
  let result = await new Thenable(1);
  alert(result);


f();
```

Error Handling

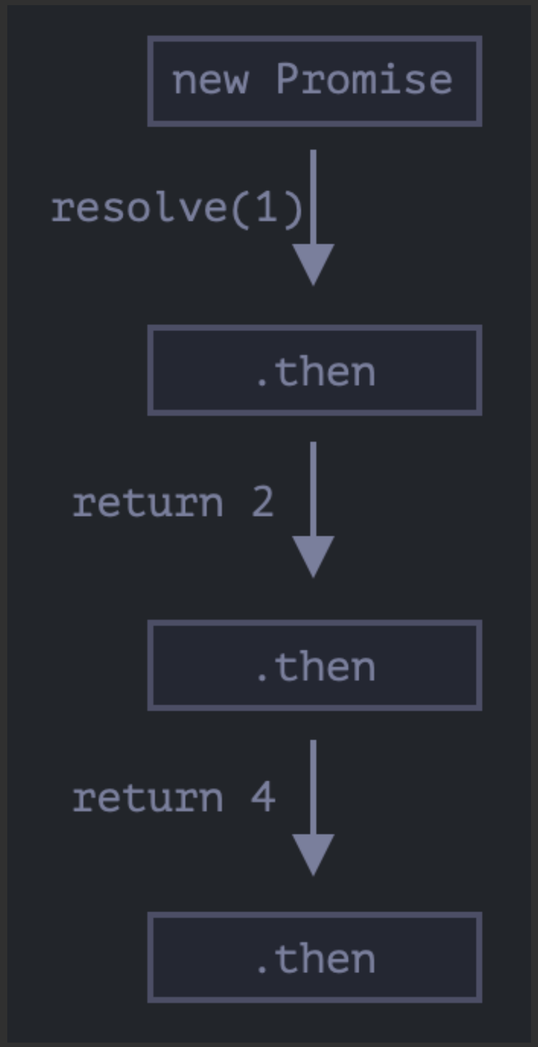
```
async function f() {
  await Promise.reject(new Error("에러 발생!"));
}


async function f() {
  throw new Error("에러 발생!");
}
```

Error Handling

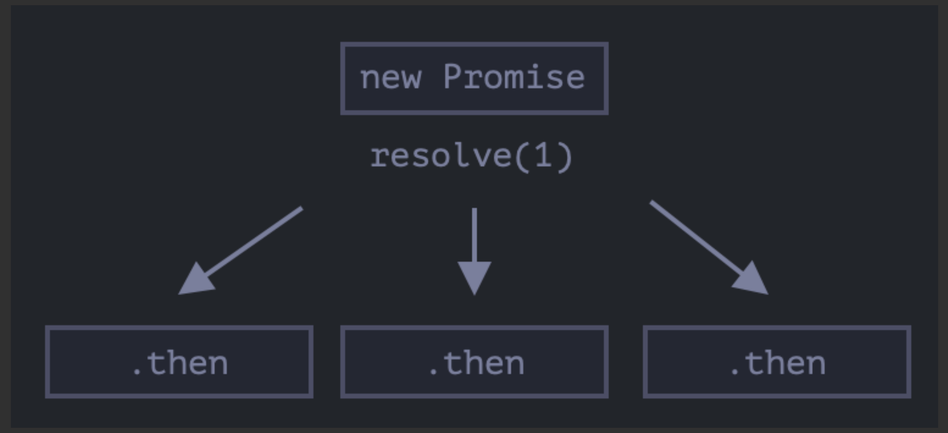
```
async function f() {

  try {
    let response = await fetch('http://유효하지-않은-주소');
    let user = await response.json();
  } catch(err) {
    // fetch와 response.json에서 발행한 에러 모두를 여기서 잡습니다.
    alert(err);
  }
}

f();
```

Google Developer Student Clubs

Promise Chaining




```
new Promise(function(resolve, reject) {

  setTimeout(() => resolve(1), 1000); // (*)

}).then(function(result) { // (**)

  alert(result); // 1
  return result * 2;

}).then(function(result) { // (***)

  alert(result); // 2
  return result * 2;

}).then(function(result) {

  alert(result); // 4
  return result * 2;

});
```




```
let promise = new Promise(function(resolve, reject) {
  setTimeout(() => resolve(1), 1000);
});

promise.then(function(result) {
  alert(result); // 1
  return result * 2;
});

promise.then(function(result) {
  alert(result); // 1
  return result * 2;
});

promise.then(function(result) {
  alert(result); // 1
  return result * 2;
});
```

```
new Promise(function(resolve, reject) {

  setTimeout(() => resolve(1), 1000);

}).then(function(result) {

  alert(result); // 1

  return new Promise((resolve, reject) => { // (*)
    setTimeout(() => resolve(result * 2), 1000);
  });

}).then(function(result) { // (**)

  alert(result); // 2

  return new Promise((resolve, reject) => {
    setTimeout(() => resolve(result * 2), 1000);
  });

}).then(function(result) {

  alert(result); // 4

});
```

Module