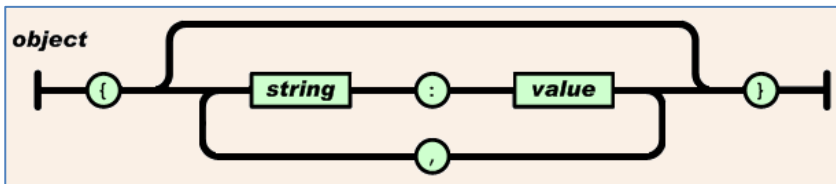
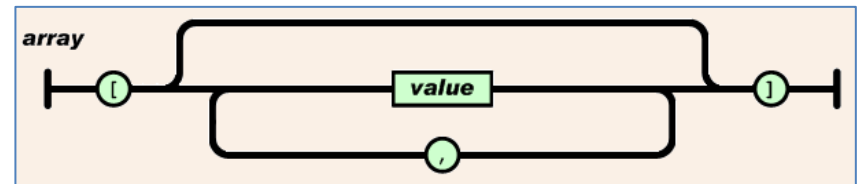


■ JSON (JavaScript Object Notation)

- JavaScript Object Notation
- 간단한 데이터 교환 형식
- 경량 텍스트 기반의 구조
- 읽기 쉬운 key : value 형식
- 특정 언어에 제한적이지 않고 독립적인 방식



```
{  
  "id" : "ggoreb",  
  "age" : 20,  
  "address" : "seoul"  
}
```

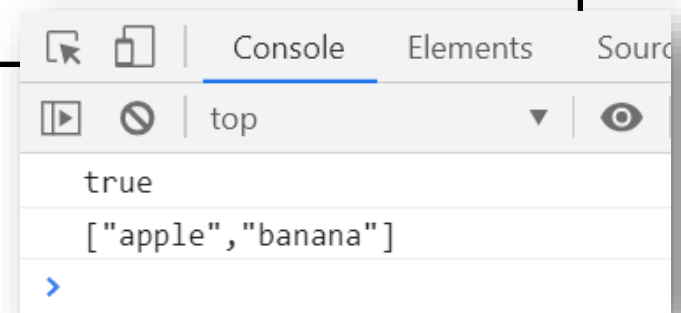


```
[  
  "seoul", "gwanak", "sillim"  
]  
  
{  
  "id" : "ggoreb",  
  "age" : 20,  
  "address" : ["seoul", "gwanak", "sillim"]  
}
```

■ JSON (JavaScript Object Notation)

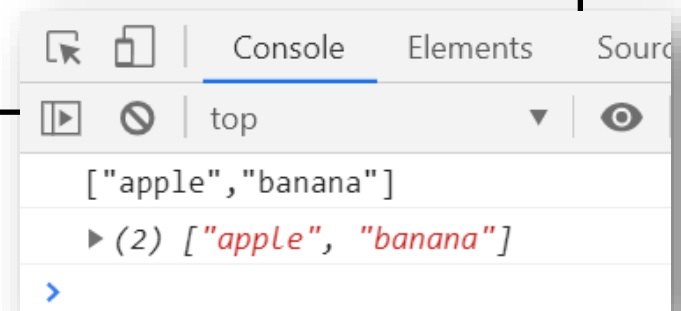
● Object → JSON (stringify)

```
let json = JSON.stringify(true);  
console.log(json);  
  
json = JSON.stringify(['apple', 'banana'])  
console.log(json);
```



● JSON → Object (parse)

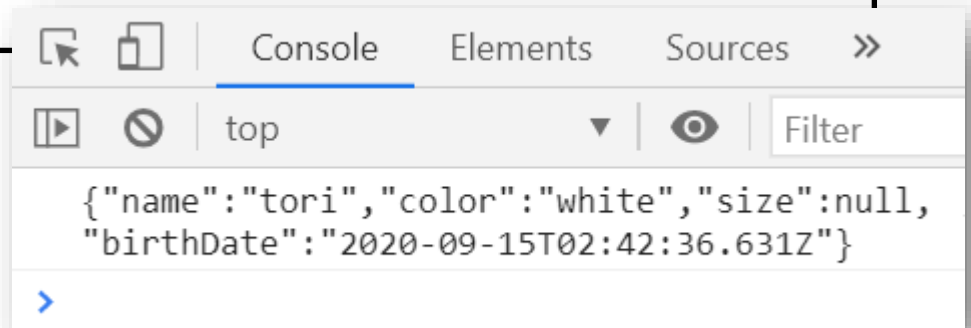
```
let json = JSON.stringify(['apple', 'banana'])  
console.log(json);  
  
const obj = JSON.parse(json);  
console.log(obj);
```



■ JSON (JavaScript Object Notation)

● JSON 생성 시 제외되는 타입

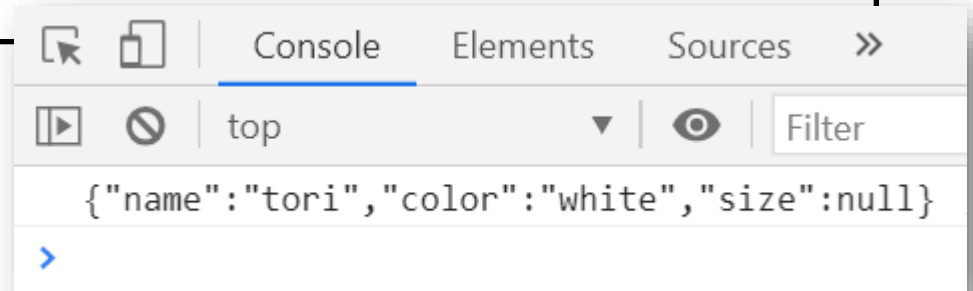
```
const rabbit = {  
  name: 'tori',  
  color: 'white',  
  size: null,  
  birthDate: new Date(),  
  jump: () => { // 함수 제외  
    console.log(`${rabbit.name} can jump!`);  
  },  
  symbol: Symbol('id') // 심볼 제외  
}  
let json = JSON.stringify(rabbit)  
console.log(json);
```



■ JSON (JavaScript Object Notation)

● JSON 생성 시 사용할 속성 지정

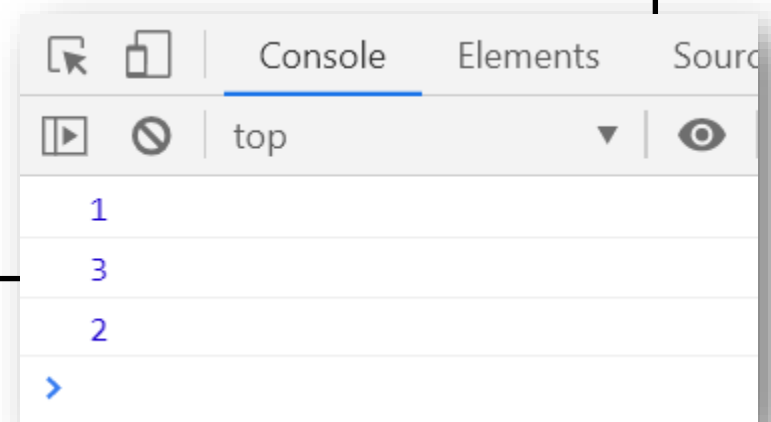
```
const rabbit = {  
  name: 'tori',  
  color: 'white',  
  size: null,  
  birthDate: new Date(),  
  jump: () => { // 함수 제외  
    console.log(`${rabbit.name} can jump!`);  
  },  
  symbol: Symbol('id') // 심볼 제외  
}  
let json = JSON.stringify(  
  rabbit, ['name', 'color', 'size'])  
console.log(json);
```



■ Callback

- 특정 시점에 호출되는 함수
- 일반적으로 함수의 매개변수로 전달하도록 설계

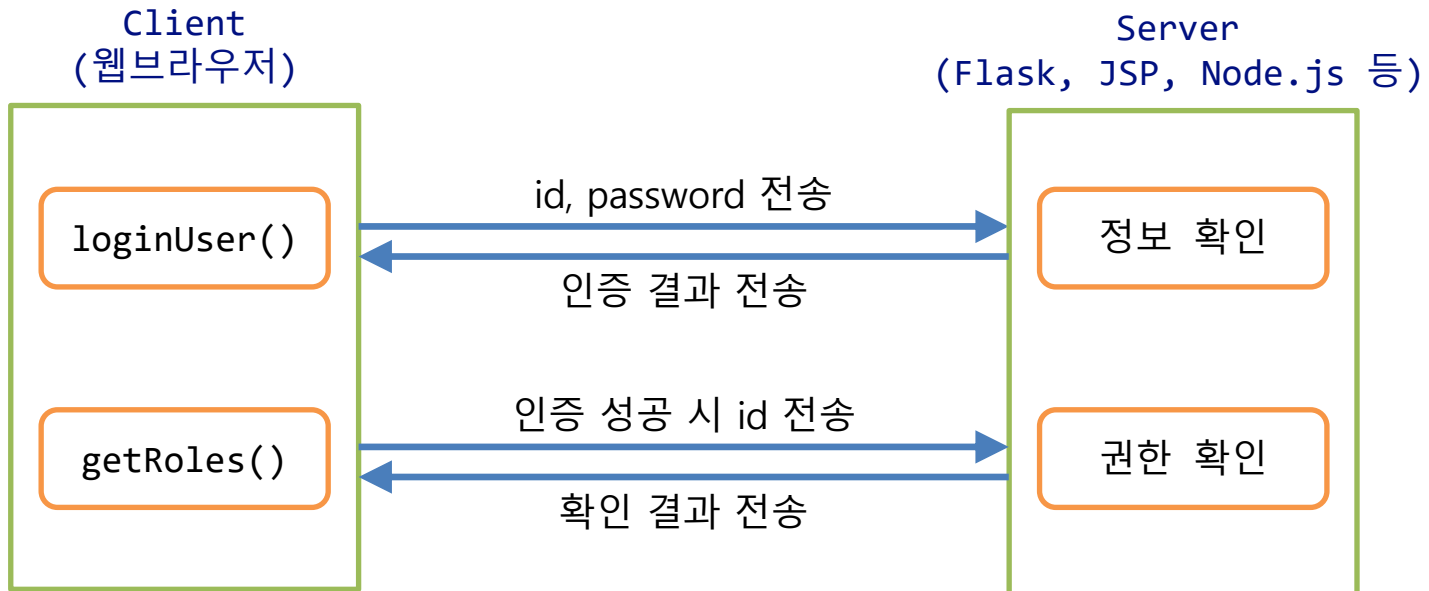
```
console.log(1);  
setTimeout(delayPrint, 1000);  
console.log(3);      Callback 함수  
  
function delayPrint() {  
    console.log(2);  
}
```



■ Callback 예시

● 로그인 후 사용자 권한 확인

```
class UserStorage {  
  login(id, password, onSuccess, onError) {  
    // id, password를 서버로 전송하여 가입정보 인증  
  }  
  
  getRoles(id, onSuccess, onError) {  
    // 로그인 성공 시 id를 서버로 전송하여 권한 확인  
  }  
}
```



■ Callback 예시

● 로그인

```
loginUser(id, password, onSuccess, onError) {  
    // id, password를 서버로 전송하여 가입정보 인증  
    setTimeout(() => {  
        if ((id === 'java' && password === 'script') ||  
            (id === 'call' && password === 'back')) {  
            onSuccess(id);  
        } else {  
            onError(new Error('not found'));  
        }  
    }, 2000);  
}
```

```
class UserStorage {  
    loginUser( ... ) {  
    }  
    getRoles( ... ) {  
    }  
}
```

■ Callback 예시

● 사용자 권한 확인

```
getRoles(id, onSuccess, onError) {  
  // 로그인 성공 시 id를 서버로 전송하여 권한 확인  
  setTimeout(() => {  
    if (id === 'java') {  
      onSuccess({ id: 'java', role: 'admin' });  
    } else if (id === 'call') {  
      onSuccess({ id: 'call', role: 'manager' });  
    } else {  
      onError(new Error('no access'));  
    }  
  }, 1000);  
}
```

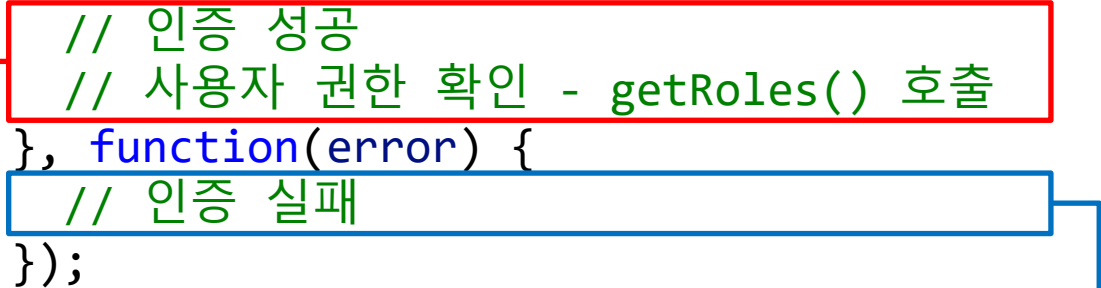
```
class UserStorage {  
  loginUser( ... ) {  
  
  }  
  getRoles( ... ) {  
  
  }  
}
```


■ Callback 예시

● 클래스 생성 및 메소드 호출 (로그인)

```
const userStorage = new UserStorage();
const id = 'java';
const password = 'script';

userStorage.loginUser(id, password, function(id) {
  // 인증 성공
  // 사용자 권한 확인 - getRoles() 호출
}, function(error) {
  // 인증 실패
});
```

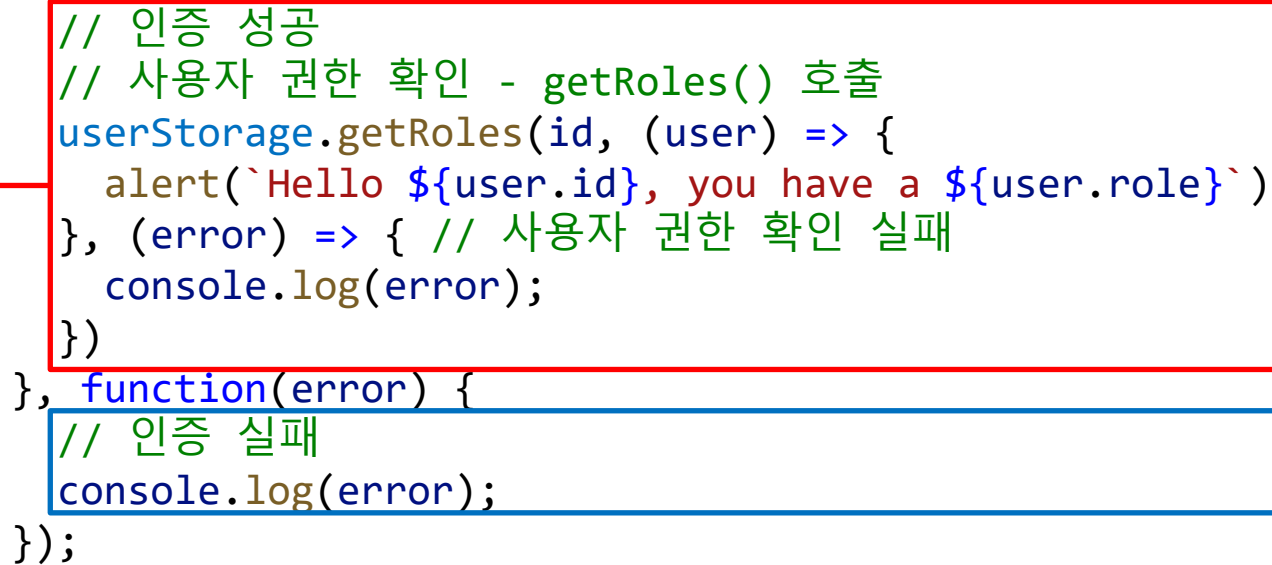


```
loginUser(id, password, onSuccess, onError) {
  // id, password를 서버로 전송하여 가입정보 인증
  setTimeout(() => {
    if ((id === 'java' && password === 'script') ||
        (id === 'call' && password === 'back')) {
      onSuccess(id);
    } else {
      onError(new Error('not found'));
    }
  }, 2000);
}
```

■ Callback 예시

● 클래스 생성 및 메소드 호출 (로그인)

```
userStorage.loginUser(id, password, function(id) {  
  // 인증 성공  
  // 사용자 권한 확인 - getRoles() 호출  
  userStorage.getRoles(id, (user) => {  
    alert(`Hello ${user.id}, you have a ${user.role}`)  
  }, (error) => { // 사용자 권한 확인 실패  
    console.log(error);  
  })  
}, function(error) {  
  // 인증 실패  
  console.log(error);  
});
```



```
loginUser(id, password, onSuccess, onError) {  
  // id, password를 서버로 전송하여 가입정보 인증  
  setTimeout(() => {  
    if ((id === 'java' && password === 'script') ||  
        (id === 'call' && password === 'back')) {  
      onSuccess(id);  
    } else {  
      onError(new Error('not found'));  
    }  
  }, 2000);  
}
```

■ Callback 예시

● Callback 함수 내에서 메소드 호출 (사용자 권한 확인)

```
userStorage.loginUser(id, password, function(id) {  
  // 인증 성공  
  // 사용자 권한 확인 - getRoles() 호출  
  userStorage.getRoles(id, (user) => {  
    alert(`Hello ${user.id}, you have a ${user.role}`)  
  }, (error) => { // 사용자 권한 확인 실패  
    console.log(error);  
  })  
}, function(error) {  
  // 인증 실패  
  console.log(error);  
});
```

```
getRoles(id, onSuccess, onError) {  
  // 로그인 성공 시 id를 서버로 전송하여 권한 확인  
  setTimeout(() => {  
    if (id === 'java') {  
      onSuccess({ id: 'java', role: 'admin' });  
    } else if (id === 'call') {  
      onSuccess({ id: 'call', role: 'manager' });  
    } else {  
      onError(new Error('no access'));  
    }  
  }, 1000);  
}
```

■ Callback 예시

● 전체 코드

```
class UserStorage {
  login(id, password, onSuccess, onError) {
    // id, password를 서버로 전송하여 가입정보 인증
    setTimeout(() => {
      if ((id === 'java' && password === 'script') ||
        (id === 'call' && password === 'back')) {
        onSuccess(id);
      } else {
        onError(new Error('not found'));
      }
    }, 2000);
  }

  getRoles(id, onSuccess, onError) {
    // 로그인 성공 시 id를 서버로 전송하여 권한 확인
    setTimeout(() => {
      if (id === 'java') {
        onSuccess({ id: 'java', role: 'admin' });
      } else if (id === 'call') {
        onSuccess({ id: 'call', role: 'manager' });
      } else {
        onError(new Error('no access'));
      }
    }, 1000);
  }
}
```

```
const userStorage = new UserStorage();
const id = 'java';
const password = 'script';

// 단점
// 가독성 떨어짐 (구조가 복잡함)
userStorage.login(id, password, function(id) {
  // 인증 성공
  // 사용자 권한 확인 - getRoles() 호출
  userStorage.getRoles(id, (user) => {
    alert(
      `Hello ${user.id}, you have a ${user.role}`
    ), (error) => { // 사용자 권한 확인 실패
      console.log(error);
    }
  })
}, function(error) {
  // 인증 실패
  console.log(error);
});
```

■ Promise

- 비동기 작업을 간편하게 처리할 수 있도록 도와주는 객체
- 콜백 함수 대신 사용할 수 있는 객체
- 정상 수행 → 성공 메시지 (resolve)
비정상 수행 → 에러 메시지 (reject)
- Promise 사용 시 알아야 할 2가지 사항
 - 상태(State)
 - 수행중 (Pending) → 완료 (Fulfilled)
 - 수행중 (Pending) → 에러 (Rejected)
 - 데이터 제공자와 데이터 사용자 (Producer / Consumer) 구분
- Promise 기본 사용법

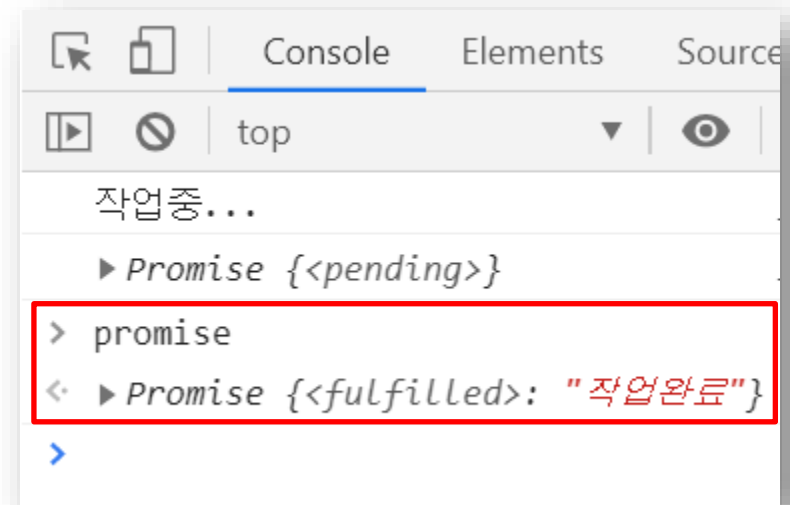
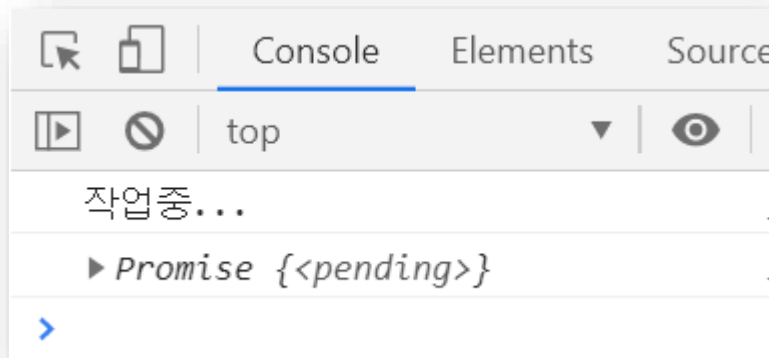
```
Promise(executor: (resolve: (value?: any) => void,  
reject: (reason?: any) => void) => void);
```

```
const promise = new Promise(function(resolve, reject) {  
  // doing some heavy work (network, read files)  
});
```

■ Promise

● Producer (데이터 제공자) - Promise 객체

```
const promise = new Promise((resolve, reject) => {  
  console.log('작업중...');  
  setTimeout(() => {  
    resolve('작업완료');  
    // reject(new Error('응답없음'));  
  }, 2000);  
});  
console.log(promise);
```

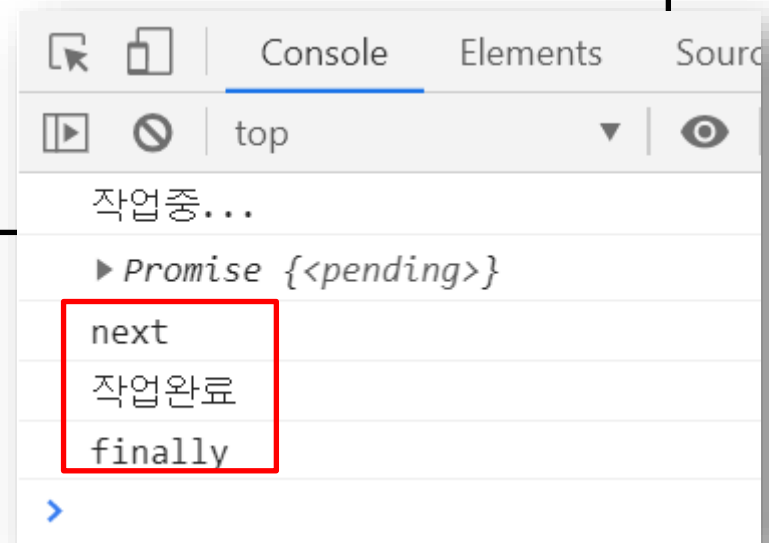


■ Promise

● Consumer (데이터 사용자) – then / catch / finally

```
promise ②
  .then((value) => { // resolve
    console.log(value);
  })
  .catch((error) => {
    console.log(error); // reject
  })
  .finally(() => { ③
    console.log('finally');
  });

console.log('next' ①);
```



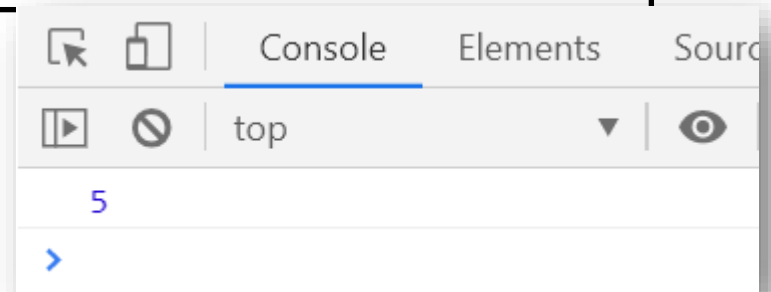
■ Promise

● Chaining

```
const fetchNumber = new Promise((resolve, reject) => {  
  setTimeout(() => resolve(1), 1000); ①  
})
```

fetchNumber

```
.then(num => num * 2) ②  
.then(num => num * 3) ③  
.then(num => { ④  
  return new Promise((resolve, reject) => {  
    setTimeout(() => resolve(num - 1)); ⑤  
  })  
})  
.then(num => console.log(num)); ⑥
```

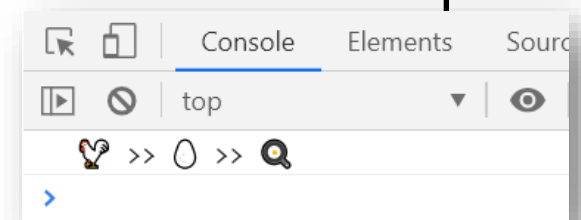


■ Promise

● Chaining

```
const getHen = () =>
  new Promise((resolve, reject) => { ②
    setTimeout(() => resolve('🐔'), 1000);
  });
const getEgg = (hen) =>
  new Promise((resolve, reject) => { ⑤
    setTimeout(() => resolve(`${hen} >> 🥚`), 1000);
  });
const cook = (egg) =>
  new Promise((resolve, reject) => { ⑧
    setTimeout(() => resolve(`${egg} >> 🍳`), 1000);
  });
```

```
getHen() ①
  .then(hen => getEgg(hen)) ③ ④
  .then(egg ⑥ => cook(egg)) ⑦
  .then(result ⑨ => console.log(result)) ⑩
```



■ Callback 예시 → Promise 적용

● 로그인 후 사용자 권한 확인

```
class UserStorage {  
  login(id, password, onSuccess, onError) {  
    // id, password를 서버로 전송하여 가입정보 인증  
  }  
  
  getRoles(id, onSuccess, onError) {  
    // 로그인 성공 시 id를 서버로 전송하여 권한 확인  
  }  
}
```



```
class UserStorage {  
  login(id, password) {  
    return new Promise((resolve, reject) => {  
      // resolve() or reject()  
    });  
  }  
  
  getRoles(id) {  
    return new Promise((resolve, reject) => {  
      // resolve() or reject()  
    });  
  }  
}
```

■ Callback 예시 → Promise 적용

● 로그인

```
loginUser(id, password, onSuccess, onError) {  
  setTimeout(() => {  
    if ((id === 'java' && password === 'script') ||  
        (id === 'call' && password === 'back')) {  
      onSuccess(id);  
    } else {  
      onError(new Error('not found'));  
    }  
  }, 1000);  
}
```



```
loginUser(id, password) {  
  return new Promise((resolve, reject) => {  
    setTimeout(() => {  
      if ((id === 'java' && password === 'script') ||  
          (id === 'call' && password === 'back')) {  
        resolve(id);  
      } else {  
        reject(new Error('not found'));  
      }  
    }, 2000);  
  }));  
}
```

■ Callback 예시 → Promise 적용

● 사용자 권한 확인

```
getRoles(id, onSuccess, onError) {  
  setTimeout(() => {  
    if (id === 'java') {  
      onSuccess({ id: 'java', role: 'admin' });  
    } else if (id === 'call') {  
      onSuccess({ id: 'call', role: 'manager' });  
    } else {  
      onError(new Error('no access'));  
    }  
  }, 1000);  
}
```



```
getRoles(id) {  
  return new Promise((resolve, reject) => {  
    setTimeout(() => {  
      if (id === 'java') {  
        resolve({ id: 'java', role: 'admin' });  
      } else if (id === 'call') {  
        resolve({ id: 'call', role: 'manager' });  
      } else {  
        reject(new Error('no access'));  
      }  
    }, 1000);  
  });  
}
```

■ Callback 예시 → Promise 적용

● 메소드 호출

```
userStorage.loginUser(id, password, function(id) { // 인증 성공
  userStorage.getRoles(id, (user) => { // 사용자 권한 확인
    alert(`Hello ${user.id}, you have a ${user.role}`)
  }, (error) => { // 사용자 권한 확인 실패
    console.log(error);
  })
}, function(error) { // 인증 실패
  console.log(error);
});
```



```
userStorage.loginUser(id, password)
  .then(id => userStorage.getRoles(id))
  .then(user => alert(`Hello ${user.id}, you have a ${user.role}`))
  .catch(result => console.log(result));
```

■ Callback 예시 → Promise 적용

● 전체 코드 (Promise 적용 전)

```
class UserStorage {
  login(id, password, onSuccess, onError) {
    setTimeout(() => {
      if ((id === 'java' && password === 'script') ||
          (id === 'call' && password === 'back')) {
        onSuccess(id);
      } else {
        onError(new Error('not found'));
      }
    }, 2000);
  }

  getRoles(id, onSuccess, onError) {
    setTimeout(() => {
      if (id === 'java') {
        onSuccess({ id: 'java', role: 'admin' });
      } else if (id === 'call') {
        onSuccess({ id: 'call', role: 'manager' });
      } else {
        onError(new Error('no access'));
      }
    }, 1000);
  }
}
```

```
const userStorage = new UserStorage();
const id = 'java';
const password = 'script';

userStorage.login(id, password, function(id) {
  userStorage.getRoles(id, (user) => {
    alert(`Hello ${user.id},
          you have a ${user.role}`)
  }, (error) => {
    console.log(error);
  })
}, function(error) {
  console.log(error);
}));
```

■ Callback 예시 → Promise 적용

● 전체 코드 (Promise 적용 후)

```
class UserStorage {
  loginUser(id, password) {
    return new Promise((resolve, reject) => {
      setTimeout(() => {
        if ((id === 'java' && password === 'script') ||
            (id === 'call' && password === 'back')) {
          resolve(id);
        } else {
          reject(new Error('not found'));
        }
      }, 1000);
    });
  }

  getRoles(id) {
    return new Promise((resolve, reject) => {
      setTimeout(() => {
        if (id === 'java') {
          resolve({ id: 'java', role: 'admin' });
        } else if (id === 'call') {
          resolve({ id: 'call', role: 'manager' });
        } else {
          reject(new Error('no access'));
        }
      }, 1000);
    });
  }
}
```

```
const userStorage = new UserStorage();
const id = 'java';
const password = 'script';

userStorage.loginUser(id, password)
  .then(id => userStorage.getRoles(id))
  .then(user =>
    alert(`Hello ${user.id},
          you have a ${user.role}`))
  .catch(result => console.log(result));
```