

7. JSON, Callback, AJAX

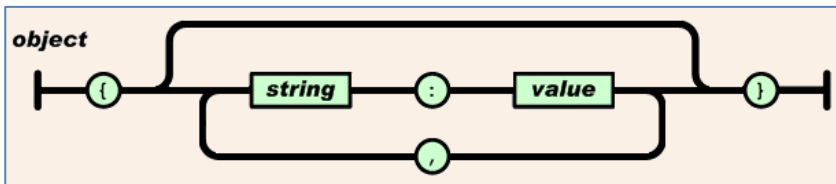
7-1. JSON

7-2. Callback

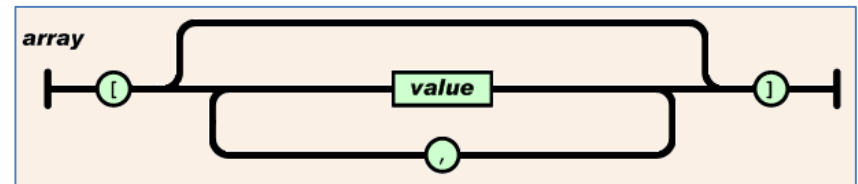
7-3. AJAX

■ 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"  
}
```



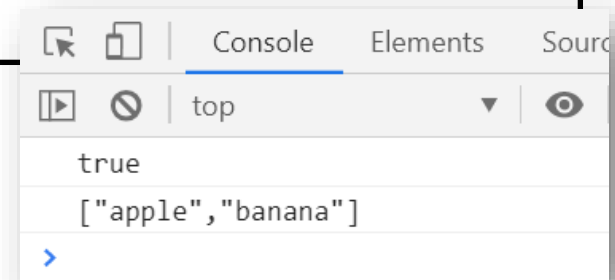
```
[  
  "seoul", "gwanak", "sillim"  
]  
  
{  
  "id" : "ggoreb",  
  "age" : 20,  
  "address" : ["seoul", "gwanak", "sillim"]  
}
```

■ JSON (JavaScript Object Notation)

● Object → JSON (stringify)

07-01-json-stringify.html

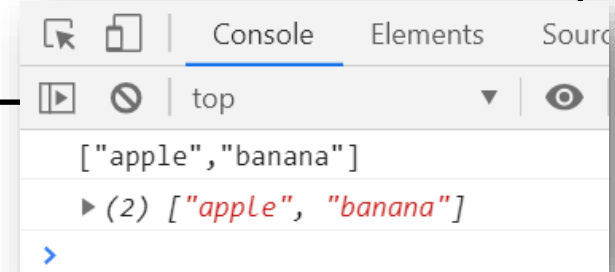
```
let json = JSON.stringify(true);  
console.log(json);  
  
json = JSON.stringify(['apple', 'banana'])  
console.log(json);
```



● JSON → Object (parse)

07-02-json-parse.html

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

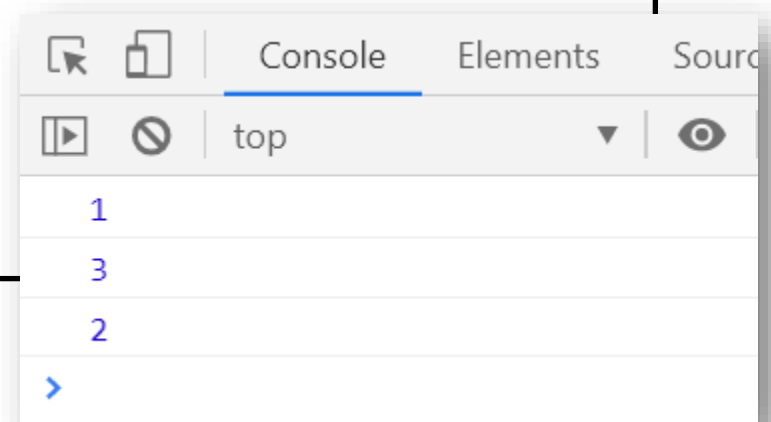


■ Callback

- 특정 시점에 호출되는 함수
- 일반적으로 함수의 매개변수로 전달하도록 설계

07-03-callback.html

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

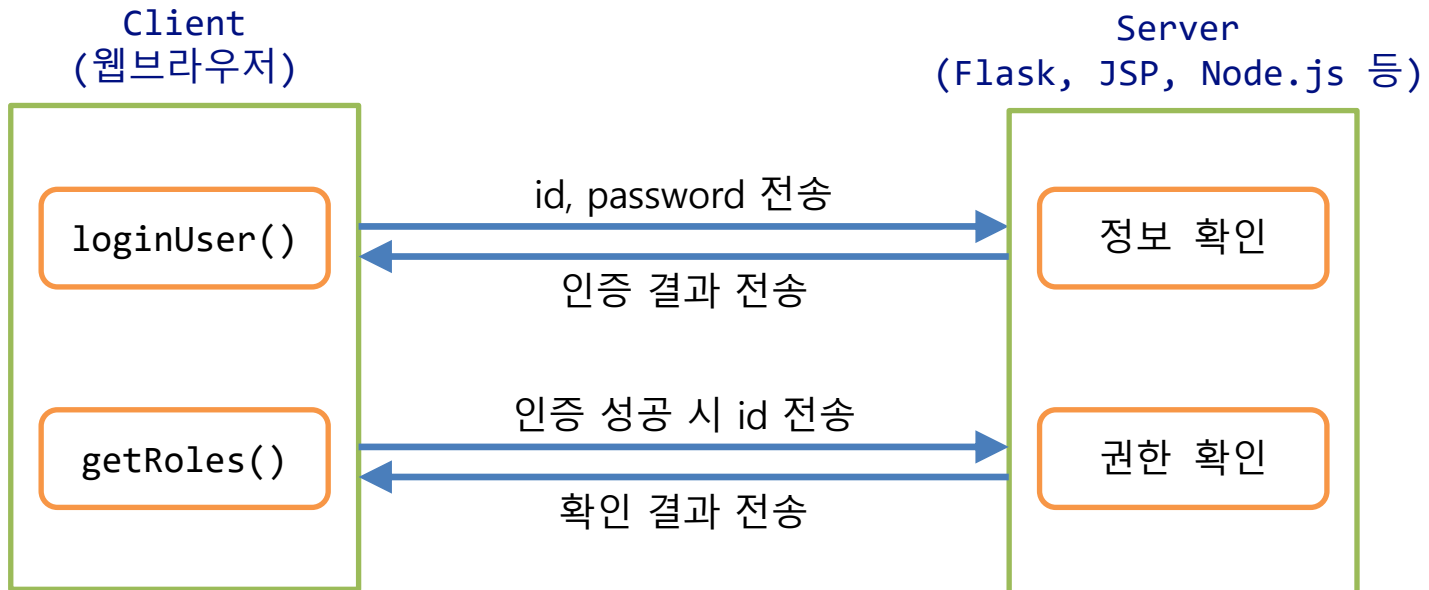


■ Callback 예시

● 로그인 후 사용자 권한 확인

07-04-user-storage.html

```
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

07-04-user-storage.html

```
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

07-04-user-storage.html

```
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( ... ) {  
  
  }  
}
```

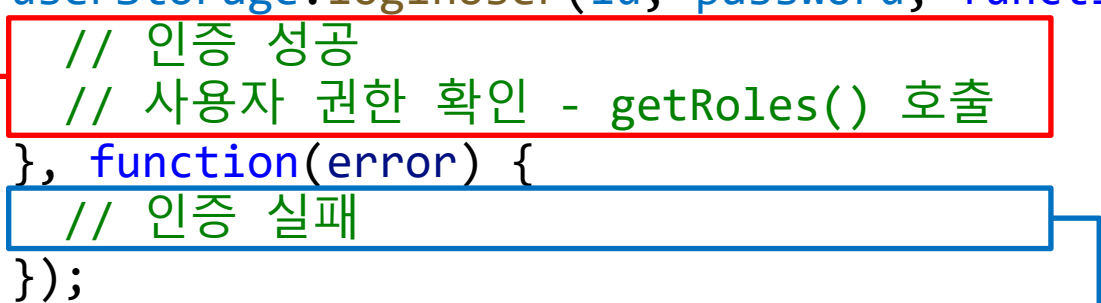
■ Callback 예시

● 클래스 생성 및 메소드 호출 (로그인)

07-04-user-storage.html

```
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)

07-04-user-storage.html

```
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)

07-04-user-storage.html

```
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

07-04-user-storage.html

```
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);
```

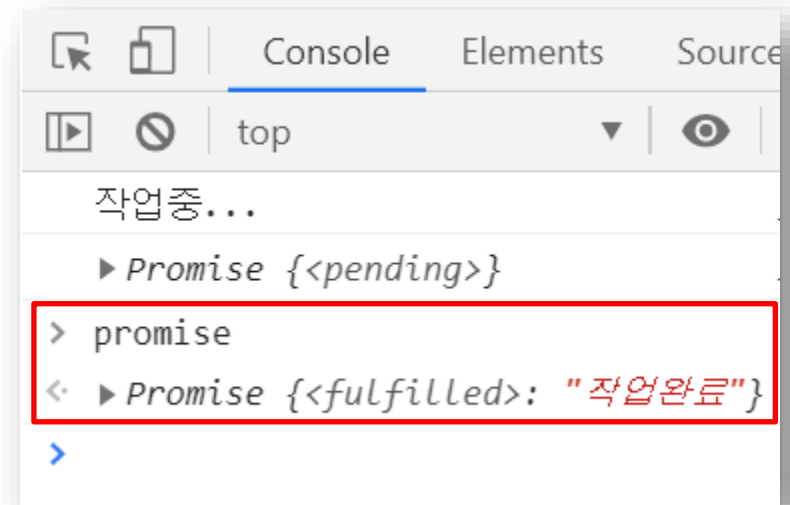
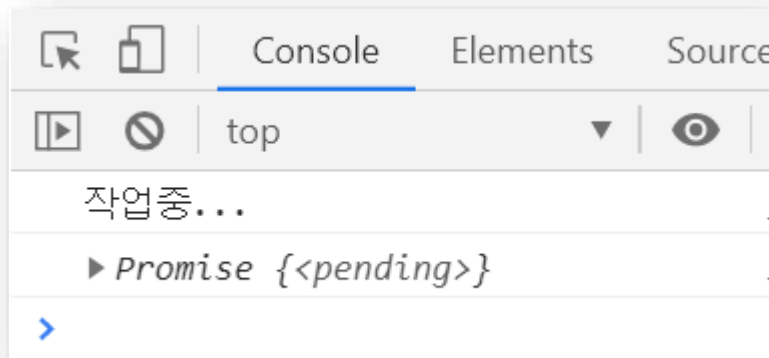
```
const promise = new Promise(function(resolve, reject) {  
  // doing some heavy work (network, read files)  
});
```

■ Promise

● Producer (데이터 제공자) - Promise 객체

07-05-promise.html

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



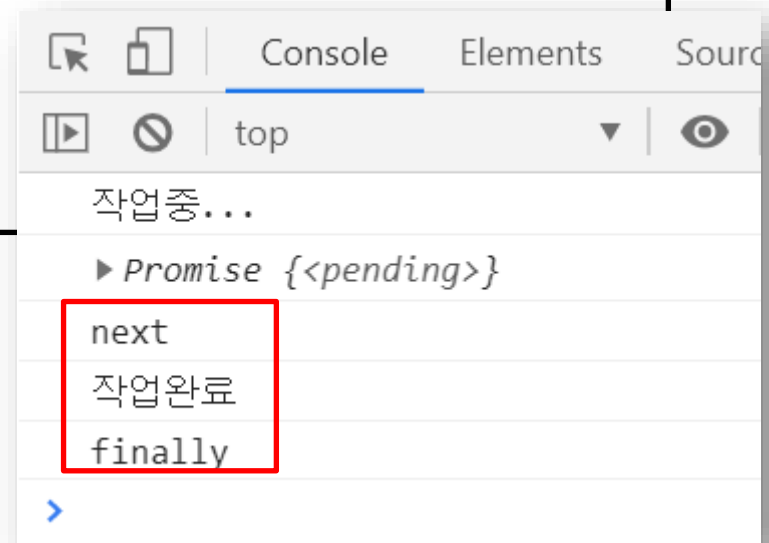
■ Promise

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

07-06-promise.html

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

console.log('next' ①);
```



■ Promise

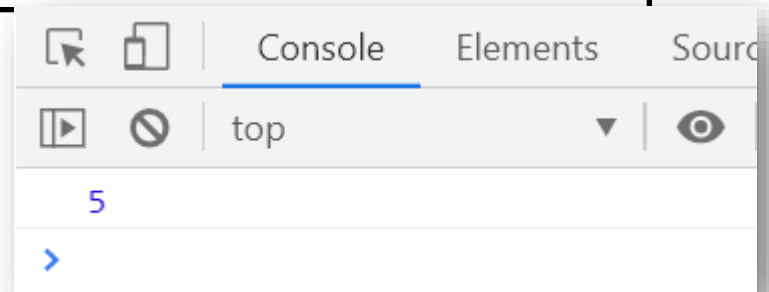
● Chaining

07-07-chaining.html

```
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) => {  
    resolve(num - 1); ⑤  
  })  
})  
.then(num => console.log(num)); ⑥
```



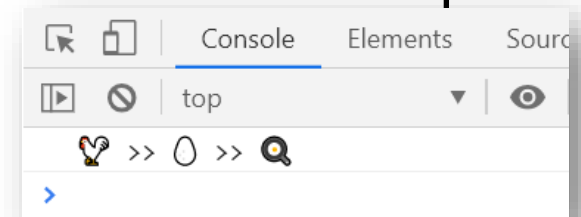
■ Promise

● Chaining

07-08-chaining.html

```
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

07-09-user-storage.html

```
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

07-09-user-storage.html

```
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

07-09-user-storage.html

```
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

07-09-user-storage.html

```
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)

07-04-user-storage.html

```
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)

07-09-user-storage.html

```
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));
```

■ AJAX

● Aynchronous JavaScript And XML

● 비동기 통신

가

● 활용 예)

- 구글 / 네이버 지도
- 유튜브 영상 재생 중 댓글 달기
- 가입을 위한 정보 입력 중 아이디 중복 검사
- 포털 사이트에서 검색어 입력 중 추천 검색어 제시 등

(Ajax)

1.
 - 가
2. fetch()
 - 2-1. fetch() await
 - 2-2. await async

// get(); // hoisting

```
async function get() {  
  const url = 'http://ggoreb.com/http/html1.jsp';  
  // response result  
  const res = await fetch(url);  
  const data = await res.text();// res.json();  
  //const data = await res.json();  
  console.log(data);  
}
```

■ AJAX

● JavaScript Old API

```
const xhr = new XMLHttpRequest();  
xhr.onreadystatechange = 콜백 함수;  
xhr.open('메소드', '주소');  
xhr.send(formData);
```

● JavaScript New API

```
const ajax = fetch('주소', { 옵션 });  
ajax.then(콜백 함수);
```

● jQuery

```
<script src='CDN 주소'></script>  
  
$.ajax({  
  url: '주소',  
  type: '메소드',  
  data: Query Parameter  
  success: 콜백 함수  
});
```

● Axios

```
<script src='CDN 주소'></script>  
  
const data = axios({  
  url: '주소',  
  method: 'get',  
  params: Query Parameter  
});  
data.then(콜백 함수);
```


■ AJAX

● JavaScript Old API (GET)

```
<script>
  const xhr = new XMLHttpRequest();
  xhr.onreadystatechange = callback;
  xhr.open('get', '주소');
  xhr.send(null);

  function callback() {
    if(xhr.readyState == 4) {
      if(xhr.status == 200) {
        let result = xhr.response;
        console.log(result);
      }
    }
  }
</script>
```

■ AJAX

● JavaScript Old API (GET)

07-10-ajax-old-get.html

```
<div id="content"></div>

<script>
  const xhr = new XMLHttpRequest();
  xhr.onreadystatechange = callback;
  xhr.open('get', 'http://ggoreb.com/ajax/html/data1.jsp');
  xhr.send(null);
  function callback() {
    if (xhr.readyState == 4) {
      if (xhr.status == 200) {
        let result = xhr.response;
        console.log(result);
        document.querySelector('#content').innerHTML += result;
      }
    }
  }
</script>
```

아이디
비밀번호

```
<div id="content">
  <h1>아이디</h1>
  <h1>비밀번호</h1>
</div>
```

■ AJAX

● JavaScript Old API (POST)

```
<script>
  const xhr = new XMLHttpRequest();
  xhr.onreadystatechange = callback;
  xhr.open('post', '주소');

  xhr.setRequestHeader('key', 'value');

  const formData = new FormData();
  formData.append('key', 'value');
  xhr.send(formData);

  function callback() {
    if (xhr.readyState == 4) {
      if (xhr.status == 200) {
        let result = xhr.response;
        console.log(result);
      }
    }
  }
</script>
```

■ AJAX

● JavaScript Old API (POST)

07-11-ajax-old-post.html

```
<div id="content"></div>

<script>
  const xhr = new XMLHttpRequest();
  xhr.onreadystatechange = callback;
  xhr.open('post', 'http://ggoreb.com/ajax/html/data2.jsp');
  xhr.send(null);
  function callback() {
    if (xhr.readyState == 4) {
      if (xhr.status == 200) {
        let result = xhr.response;
        console.log(result);
        document.querySelector('#content').innerHTML += result;
      }
    }
  }
</script>
```

- A
- B

```
<div id="content">
  <ul>
    <li>...</li>
    <li>...</li>
  </ul>
</div>
```

■ AJAX

● JavaScript New API (GET)

```
<script>
  const ajax = fetch('주소');
  ajax.then(function(response) {
    return response.text();
  }).then(function(result) {
    console.log(result);
  });
</script>
```

■ AJAX

● JavaScript New API (GET)

07-12-ajax-new-get.html

```
<div id="content"></div>

<script>
  const ajax = fetch('http://ggoreb.com/ajax/html/data1.jsp');
  ajax.then(res => {
    return res.text();
  }).then(result => {
    console.log(result);
    document.querySelector('#content').innerHTML += result;
  });
</script>
```

아이디
비밀번호

```
▼ <div id="content">
  <h1>아이디</h1>
  <h1>비밀번호</h1>
</div>
```

■ AJAX

● JavaScript New API (POST - Form Data)

```
<script>
  const formData = new FormData();
  formData.append('key', 'value');
  const ajax = fetch('주소', {
    method: 'post', body: formData
  });
  ajax.then(function(response) {
    return response.text();
  }).then(function(result) {
    console.log(result);
  });
</script>
```

■ AJAX

● JavaScript New API (POST)

07-13-ajax-new-post.html

```
<div id="content"></div>

<script>
  const formData = new FormData();
  const ajax = fetch(
    'http://ggoreb.com/ajax/html/data2.jsp',
    { method: 'post', body: formData }
  );
  ajax.then(res => {
    return res.text();
  }).then(result => {
    console.log(result);
    document.querySelector('#content').innerHTML += result;
  });
</script>
```

- A
- B

```
<div id="content">
  <ul>
    <li>...</li>
    <li>...</li>
  </ul>
</div>
```


■ AJAX

● JavaScript New API (POST - JSON Data)

```
<script>
  const formData = new FormData();
  formData.append('key', 'value');

  const ajax = fetch('주소', {
    method: 'post',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify(Object.fromEntries(formData))
  });
  ajax.then(function(response) {
    return response.json();
  }).then(function(result) {
    console.log(result);
  });
</script>
```

■ AJAX

● Axios (GET)

```
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
<script>
  const data = axios({
    url: '주소',
    method: 'get',
    params: {
      'key': 'value'
    }
  });
  data.then(function (result) {
    console.log(result.data);
  });
</script>
```

■ AJAX

● Axios (POST)

```
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
<script>
    const formData = new FormData();
    formData.append('key', 'value');

    const data = axios({
        url: '주소',
        method: 'post',
        data: formData,
        headers: {
            'key': 'value'
        }
    });
    data.then(function (result) {
        console.log(result.data);
    });
</script>
```

■ AJAX

● 언어 및 설명 데이터

07-14-ajax-axios-toc.html

```
const ajax = axios({
  url: 'http://ggoreb.com/api/toc.jsp',
  method: 'get',
  params: {}
});

ajax.then(res => {
  let code = '<ul class="list-group">';
  for(const item of res.data) {
    code += `<li
      class="list-group-item d-flex justify-content-between
      align-items-center"
      title="${item.desc}">
      ${item.title}
      <span class="badge bg-primary rounded-pill">${item.id}</span>
    </li>`;
  }
  code += '</ul>';
  document.querySelector('#toc').innerHTML = code;
});
```

HTML

1

CSS

2

JavaScript

3

React

4

■ AJAX

● 공통 데이터

07-15-ajax-axios-dinosaur.html

```
const ajax = axios({
  url: 'http://ggoreb.com/dinosaur/info.jsp',
  method: 'get',
  params: {}
});

ajax.then(res => {
  let code = '';
  for(const item of res.data) {
    code += `<div class="dropdown">
      <button class="btn btn-primary dropdown-toggle" data-bs-toggle="dropdown">
        ${item.kind} (${item['dinosaurs'].length})
      </button>`;
    code += `<ul class="dropdown-menu">`;
    for(const dinosaur of item['dinosaurs']) {
      code += `<li><a class="dropdown-item" href="#">${dinosaur.title}</a></li>`;
    }
    code += `</ul></div>`;
  }
  document.querySelector('#content').innerHTML = code;
});
```

