8. 비동기처리

8-1. async

8-2. await

■ async / await

- 비동기 작업을 동기적으로 실행되는 것처럼 보이게 만드는 방법
- promise – then – then 방식을 간단하게 표현하는 방법 **08-01-async-await.html**

```
function fetchUser() {
  console.log('작업중...');
  return new Promise((resolve, reject) => {
    resolve('작업완료');
  });
}
const user = fetchUser();
user.then(result => console.log(result))
```

⬇

```
async function fetchUser() {
  console.log('작업중...');
  return '작업완료';
}
const user = fetchUser();
user.then(result => console.log(result))
```

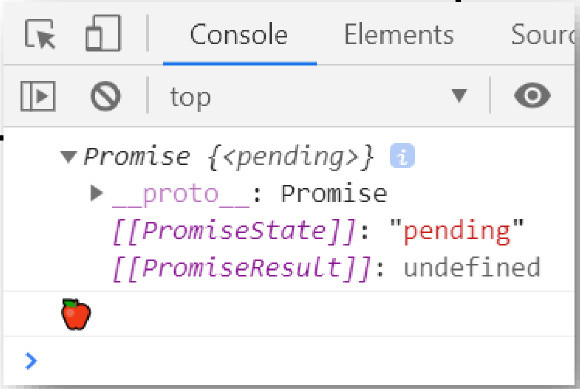
■ async / await

● '사과' 를 1초 동안 가져와서 판매하는 코드 작성

- Promise / then

08-02-promise-then.html

```
function getApple() {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve('🍎');
    }, 1000)
  });
}
function sell() {
  return new Promise((resolve) => {
    getApple().then(apple => resolve(apple));
  });
}
sell().then(res => console.log(res));
console.log(sell());  // 사용 불가
```



```
▼Promise {<pending>}
  ▶__proto__: Promise
   [[PromiseState]]: "pending"
   [[PromiseResult]]: undefined
🍎
>
```

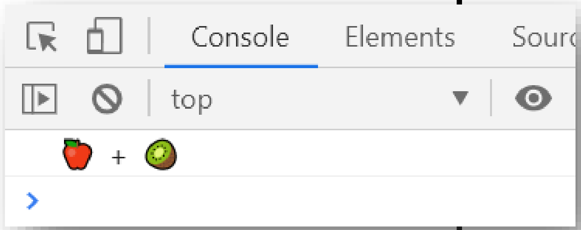
■ async / await

● **<u>'사과' 와 '키위' 를 각각 1초 동안</u>** 가져와서 판매하는 코드 작성

- Promise / then

08-03-promise-then.html

```
function getApple() {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve('🍎');
    }, 1000)
  });
}
function getKiwi() {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve('🥝');
    }, 1000)
  });
}
function sell() {
  return getApple().then(apple => {
    return getKiwi().then(kiwi => `${apple} + ${kiwi}`);
  });
}
sell().then(res => console.log(res));
```



```
🍎 + 🥝
>
```

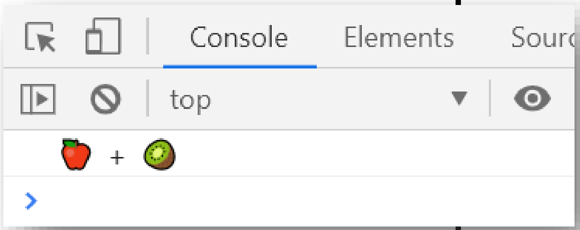
■ async / await

● **<u>'사과' 와 '키위' 를 각각 1초 동안</u>** 가져와서 판매하는 코드 작성

- async / await 적용

08-04-async-await.html

```
function getApple() {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve('🍎');
    }, 1000)
  });
}
function getKiwi() {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve('🥝');
    }, 1000)
  });
}
async function sell() {
  const apple = await getApple();
  const kiwi = await getKiwi();
  return `${apple} + ${kiwi}`
}
sell().then(res => console.log(res));
```



```
🍎 + 🥝
>
```

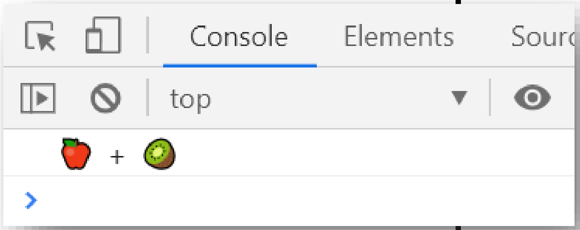
■ async / await

● **<u>'사과' 와 '키위' 를 각각 1초 동안</u>** 가져와서 판매하는 코드 작성

- setTimeout() 기능을 delay() 함수로 모듈화
- 중복코드 제거

08-05-async-await.html

```
function delay(ms) {
  return new Promise(resolve => setTimeout(resolve, ms));
}
async function getApple() {
  await delay(1000);
  return '🍎';
}
async function getKiwi() {
  await delay(1000);
  return '🥝';
}
async function sell() {
  const apple = await getApple();
  const kiwi = await getKiwi();
  return `${apple} + ${kiwi}`
}
sell().then(res => console.log(res));
```



```
🍎 + 🥝
>
```

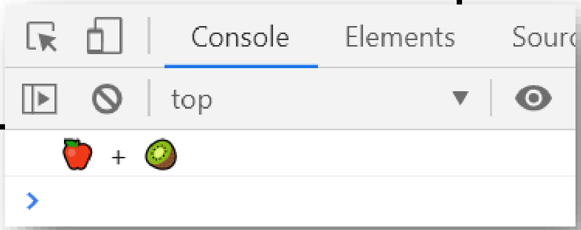
■ async / await

● **'사과' 와 '키위' 를 동시에 1초 동안(병렬)** 가져와서 판매하는 코드 작성

- 함수는 즉시 실행하고 데이터를 가져오는 동안 대기

08-06-async-await.html

```
async function sell() {
  const applePromise = getApple();
  const kiwiPromise = getKiwi();
  const apple = await applePromise;
  const kiwi = await kiwiPromise;
  return `${apple} + ${kiwi}`;
}
```



```
🍎 + 🥝
>
```

■ Promise - then → async - await

07-06-promise.html

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

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



console.log('next');
```

08-07-promise변환.html

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

const run = async () => {
  try {
    const value = await promise;
    console.log(value);
  } catch(error) {
    console.log(error);
  } finally {
    console.log('finally');
  };
};

run();

console.log('next');
```

■ Promise - then → async - await

07-07-chaining.html

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

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

08-08-promise변환.html

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

const run = async () => {
  let num = await fetchNumber;
  num = num * 2;
  num = num * 3;
  return num - 1;
}

run().then(num => console.log(num));
```

■ Promise - then → async - await

07-08-chaining.html

```
const getHen = () =>
  new Promise((resolve, reject) => {
    setTimeout(
      () => resolve('🐓'), 1000);
  });

const getEgg = (hen) =>
  new Promise((resolve, reject) => {
    setTimeout(
      () => resolve(`${hen} => 🥚`), 1000);
  });

const cook = (egg) =>
  new Promise((resolve, reject) => {
    setTimeout(
      () => resolve(`${egg} => 🍳`), 1000);
  });



getHen()
  .then(hen => getEgg(hen))
  .then(egg => cook(egg))
  .then(result => console.log(result));
```

08-09-promise변환.html

```
const delay = (ms) => {
  return new Promise(
    resolve => setTimeout(resolve, ms));
};

const getHen = async () => {
  await delay(1000);
  return '🐓';
};

const getEgg = async (hen) => {
  await delay(1000);
  return `${hen} => 🥚`;
};

const cook = async (egg) => {
  await delay(1000);
  return `${egg} => 🍳`;
}

const run = async () => {
  const hen = await getHen();
  const egg = await getEgg(hen);
  const result = await cook(egg);
  return result;
};
run().then((result) => {
  console.log(result);
});
```

■ Promise - then → async - await

07-09-user-storage.html

```
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);
  });
}
```

08-10-promise변환.html

```
async loginUser(id, password) {
  let result;
  if ((id === 'java' && password === 'script') ||
    (id === 'call' && password === 'back')) {
    result = id;
  } else {
    throw new Error('not found');
  }
  return result;
}


async getRoles(id) {
  let result;
  if (id === 'java') {
    result = { id: 'java', role: 'admin' };
  } else if (id === 'call') {
    result = { id: 'call', role: 'manager' };
  } else {
    throw new Error('no access');
  }
  return result;
}
```

■ Promise - then → async - await

07-09-user-storage.html

```
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));
```

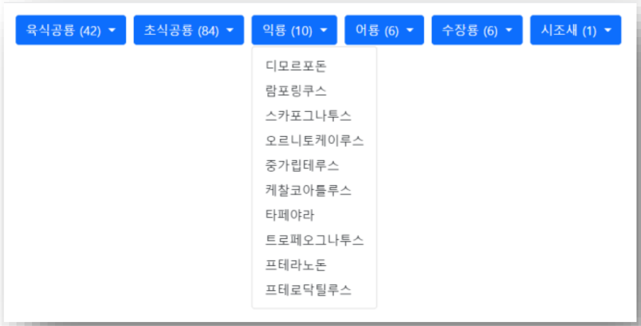
08-10-promise변환.html

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

const run = async () => {
  try {
    const userId =
      await userStorage.loginUser(id, password);
    const user = await userStorage.getRoles(userId);
    alert(`Hello ${user.id}, you have a ${user.role}`);
  } catch(result) {
    console.log(result);
  }
};

run();
```

■ 연습문제 (08-연습문제.html)

- "07-15-ajax-axios-dinosaur.html"의 코드에 async와 await를 적용해서 동일한 결과가 출력될 수 있도록 수정하기

```
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;
});
```


육식공룡 (42)
초식공룡 (84)
익룡 (10)
어룡 (6)
수장룡 (6)
시조새 (1)
디모르포돈
람포링쿠스
스카포그나투스
오르니토케이루스
중가립테루스
케찰코아틀루스
타페야라
트로페오그나투스
프테라노돈
프테로닥틸루스