9일간의 자바스크립트

비동기 복습
콜백

- setTimeout말고 브라우저의 함수를 좀 사용해 볼 겁니다.
- 이를 사용해서 콜백의 문제점이 무엇인지 알아봅시다.

비동기 복습

콜백

```
function loadScript(src) {
  const script = document.createElement("script");
  script.src = src;
  document.body.append(script);
}

// loadScript는 비동기로 작동
loadScript("./makeFunction.js");

// 존재하지 않는 함수여서 에러가 뜸
make();
```

```
function loadScript(src, callback) {
  const script = document.createElement("script");
  script.src = src;
  document.body.append(script);

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

// loadScript는 비동기로 작동
loadScript("./makeFunction.js", (script) => {
  console.log(`${script.src} is loaded!`);

  // 잘 실행이 된다.
  make();
});
```

비동기 복습

콜백

```
function loadScript(src, callback) {
  const script = document.createElement("script");
  script.src = src;
  document.body.append(script);

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

// loadScript는 비동기로 작동
loadScript("./makeFunction.js", (script) => {
  console.log(`${script.src} is loaded!`);

  // 잘 실행이 된다.
  make();
});
```

```
function loadScript(src, callback) {
  const script = document.createElement("script");
  script.src = src;
  document.body.append(script);

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

// loadScript는 비동기로 작동
loadScript("./makeFunction.js", (script) => {
  console.log(`${script.src} is loaded!`);

  loadScript("./option.js", (script) => {
    console.log(`${script.src} is loaded!`);

    make(option);
  });
});
```

비동기 복습

콜백

```
function loadScript(src, callback) {
  const script = document.createElement("script");
  script.src = src;
  document.body.append(script);

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

// loadScript는 비동기로 작동
loadScript("./makeFunction.js", (script) => {
  console.log(`${script.src} is loaded!`);

  loadScript("./option.js", (script) => {
    console.log(`${script.src} is loaded!`);

    make(option);
  });
});
```

```
function loadScript(src, callback, error) {
  const script = document.createElement("script");
  script.src = src;
  document.body.append(script);

  script.onload = () => callback(script);
  script.onerror = (err) => error("can't load script", err);
}

// loadScript는 비동기로 작동
loadScript("./makeFunction.js", (script) => {
  console.log(`${script.src} is loaded!`);

  loadScript("./option.js", (script) => {
    console.log(`${script.src} is loaded!`);

    make(option);
  }, (error, msg) => {
    console.error(error, msg);
  });
}, (error, msg) => {
  console.error(error, msg);
});
```

비동기 복습

콜백

```
function loadScript(src, callback, error) {
  const script = document.createElement("script");
  script.src = src;
  document.body.append(script);

  script.onload = () => callback(script);
  script.onerror = (err) => error("can't load script", err);
}

// loadScript는 비동기로 작동
loadScript("./makeFunction.js", (script) => {
  console.log(`${script.src} is loaded!`);

  loadScript("./option.js", (script) => {
    console.log(`${script.src} is loaded!`);

    make(option);
  }, (error, msg) => {
    console.error(error, msg);
  });
}, (error, msg) => {
  console.error(error, msg);
});
```

```
function loadScript(src, callback, error) {
  const script = document.createElement("script");
  script.src = src;
  document.body.append(script);

  script.onload = () => callback(script);
  script.onerror = (err) => error("can't load script", err);
}

// loadScript는 비동기로 작동
loadScript("./makeFunction.js", (makeScript) => {
  console.log(`${makeScript.src} is loaded!`);

  loadScript("./option.js", (optionScript) => {
    console.log(`${optionScript.src} is loaded!`);

    loadScript("./arguments.js", (argumentScript) => {
      console.log(`${argumentScript.src} is loaded!`);

      make(option, ...arguments);
    }, (error, msg) => {
      console.error(error, msg);
    });
  }, (error, msg) => {
    console.error(error, msg);
  });
}, (error, msg) => {
  console.error(error, msg);
});
```

비동기 복습

콜백

```
function loadScript(src, callback, error) {
  const script = document.createElement("script");
  script.src = src;
  document.body.append(script);

  script.onload = () => callback(script);
  script.onerror = (err) => error("can't load script", err);
}

// loadScript는 비동기로 작동
loadScript("./makeFunction.js", (makeScript) => {
  console.log(`${makeScript.src} is loaded!`);

  loadScript("./option.js", (optionScript) => {
    console.log(`${optionScript.src} is loaded!`);

    loadScript("./arguments.js", (argumentScript) => {
      console.log(`${argumentScript.src} is loaded!`);

      make(option, ...arguments);
    }, (error, msg) => {
      console.error(error, msg);
    });
  }, (error, msg) => {
    console.error(error, msg);
  });
}, (error, msg) => {
  console.error(error, msg);
});
```

callback hell

비동기 복습

Promise

```
function loadScript(src, callback, error) {
  const script = document.createElement("script");
  script.src = src;
  document.body.append(script);

  script.onload = () => callback(script);
  script.onerror = (err) => error("can't load script", err);
}

// loadScript는 비동기로 작동
loadScript("./makeFunction.js", (makeScript) => {
  console.log(`${makeScript.src} is loaded!`);

  loadScript("./option.js", (optionScript) => {
    console.log(`${optionScript.src} is loaded!`);

    loadScript("./arguments.js", (argumentScript) => {
      console.log(`${argumentScript.src} is loaded!`);

      make(option, ...arguments);
    }, (error, msg) => {
      console.error(error, msg);
    });
  }, (error, msg) => {
    console.error(error, msg);
  });
}, (error, msg) => {
  console.error(error, msg);
});
```

```
function loadScript(src) {
  return new Promise((resolve, reject) => {
    const script = document.createElement("script");
    script.src = src;
    document.body.append(script);

    script.onload = () => resolve(script);
    script.onerror = (err) => reject("can't load script", err);
  });
}
```

비동기 복습

Promise

```
function loadScript(src) {
  return new Promise((resolve, reject) => {
    const script = document.createElement("script");
    script.src = src;
    document.body.append(script);

    script.onload = () => resolve(script);
    script.onerror = (err) => reject("can't load script", err);
  });
}
```

```
function loadScript(src) {
  return new Promise((resolve, reject) => {
    const script = document.createElement("script");
    script.src = src;
    document.body.append(script);

    script.onload = () => resolve(script);
    script.onerror = (err) => reject("can't load script", err);
  });
}

loadScript("./makeFunction.js")
  .then((makeScript) => {
    console.log(`${makeScript.src} is loaded!`);

    loadScript("./option.js")
      .then((optionScript) => {
        console.log(`${optionScript.src} is loaded!`);

        loadScript("./argument.js")
          .then((argumentScript) => {
            console.log(`${argumentScript.src} is loaded!`);

            make(option, ...arguments);
          });
      });
  });
```

비동기 복습

Promise - return

```
function loadScript(src) {
  return new Promise((resolve, reject) => {
    const script = document.createElement("script");
    script.src = src;
    document.body.append(script);

    script.onload = () => resolve(script);
    script.onerror = (err) => reject("can't load script", err);
  });
}

loadScript("./makeFunction.js")
  .then((makeScript) => {
    console.log(`${makeScript.src} is loaded!`);

    loadScript("./option.js")
      .then((optionScript) => {
        console.log(`${optionScript.src} is loaded!`);

        loadScript("./argument.js")
          .then((argumentScript) => {
            console.log(`${argumentScript.src} is loaded!`);

            make(option, ...arguments);
          });
      });
  });
```

```
function loadScript(src) {
  return new Promise((resolve, reject) => {
    const script = document.createElement("script");
    script.src = src;
    document.body.append(script);

    script.onload = () => resolve(script);
    script.onerror = (err) => reject("can't load script", err);
  });
}

loadScript("./makeFunction.js")
  .then((script) => {
    console.log(`${script.src} is loaded!`);
    return loadScript("./option.js");
  })
  .then((script) => {
    console.log(`${script.src} is loaded!`);
    return loadScript("./argument.js");
  })
  .then((script) => {
    console.log(`${script.src} is loaded!`);
    make(option, ...arguments);
  });
```

비동기 복습

Promise - return

```
function loadScript(src) {
  return new Promise((resolve, reject) => {
    const script = document.createElement("script");
    script.src = src;
    document.body.append(script);

    script.onload = () => resolve(script);
    script.onerror = (err) => reject("can't load script", err);
  });
}

loadScript("./makeFunction.js")
  .then((script) => {
    console.log(`${script.src} is loaded!`);
    return loadScript("./option.js");
  })
  .then((script) => {
    console.log(`${script.src} is loaded!`);
    return loadScript("./argument.js");
  })
  .then((script) => {
    console.log(`${script.src} is loaded!`);
    make(option, ...arguments);
  });
```

then chaining

fetch

- 서버에 요청을 보내는 역할을 한다.
- fetch("https://~~")
- Promise객체를 돌려준다. => then으로 이어나갈 수 있음
- response.text().then()
- response.json().then()

fetch

- 가짜로 만든 서버에서 정보를 얻어오기!
- json이라는 데이터를 받을 것인데, object로 반환된다.

댄댄댄댄댄...

- then을 계속 쓰는 것이 불필요하다고 느껴, 새로운 문법, await이 생겼다.
- 그냥 코드를 보면서 이해하는 것이 좋을 것 같다.
- await를 사용하려면 async라는 키워드가 필요하다.

async/await

- 결과들을 모아서 보여줄 때 유리하다.
- 다만, 두가지 이상을 동시에 처리할 때는 async, await만 가지고는 무리가 있다.
  - 콜백 사용
  - Promise.all 사용

async에 대한 오해

- async 키워드는 비동기 함수를 만드는 키워드가 아닌 await키워드를 쓰기 위한 키워드다.
- 예시를 보면서~

아마도 이쯤 오면 ㄱㅣㅡㅌ