JavaScript

Part II

목차

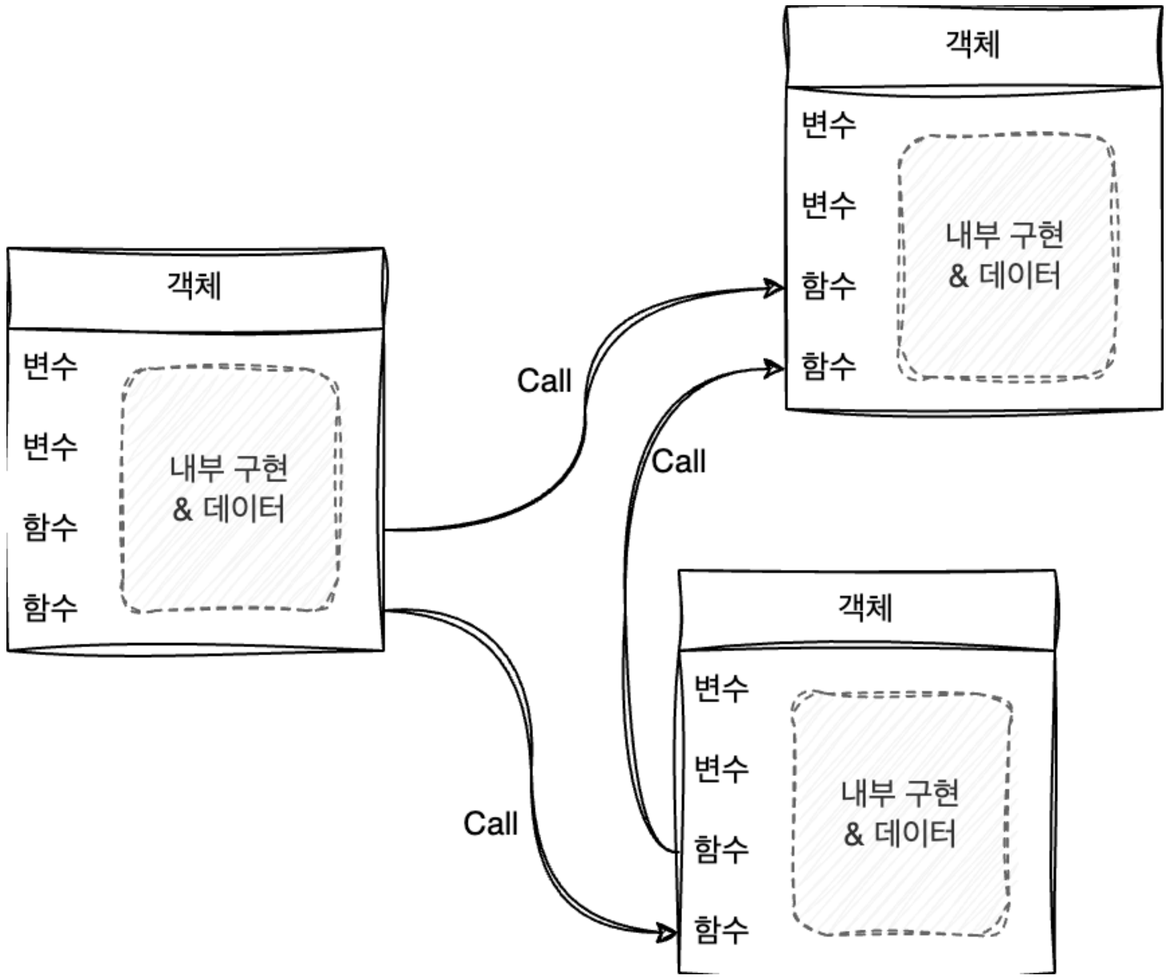
객체지향 프로그래밍

객체 지향 프로그래밍 (OOP)은 소프트웨어를 개발할 때 사용되는 프로그래밍 패러다임 중 하나로, 데이터와 해당 데이터를 조작하는 메소드를 하나의 단위인 객체(object)로 묶어서 관리하는 방식

OOP의 핵심 개념

- **추상화**: 복잡한 시스템에서 중요한 부분만 추출하여 단순화
- **캡슐화**: 객체가 자신의 데이터와 동작을 숨기고 외부에 노출하지 않도록 함
- **상속**: 부모 클래스의 속성과 메소드를 자식 클래스가 물려받아 재사용
- **다형성**: 같은 이름의 메소드가 다른 동작을 수행

객체지향 프로그래밍


객체
변수
변수
함수
함수
내부 구현
& 데이터
Call
객체
변수
변수
함수
함수
내부 구현
& 데이터
Call
객체
변수
변수
함수
함수
내부 구현
& 데이터
Call

Thaw Weight

무게 해동하기

대기모드에서 **[무게해동]** 버튼 선택후, **[시간/중량]** 눌러서 중량 "200g" 선택후, **[메뉴/시간]** 다이얼을 누르면 조리가 시작됩니다.

Cook Quickly

신속하게 조리하기

대기모드에서 **[조리시작/+30초]** 버튼을 누르면 작동됩니다.

(※ 출력이 100%이므로 5분이내로 조리를 권장합니다.)

Microwave Cooking 800W

출력 레벨 조리하기

대기모드에서 **[출력]** 버튼을 2번 누르면 " 80%", **[메뉴/시간]** 다이얼를 시계방향으로 돌려서 시간을 "60"초 선택하고, **[메뉴/시간]** 다이얼을 **누르면**, 조리가 시작됩니다.

클래스와 객체 생성

클래스는 객체를 만들기 위한 **템플릿**으로, 클래스는 **데이터를 담는 변수**와 메소드를 포함

클래스 = 붕어빵 틀, 객체 = 붕어빵

객체는 클래스를 통해 생성되며, 생성자 함수를 호출하여 객체를 초기화

JavaScript 에서는 객체의 속성과 메소드를 점(.) 연산자를 사용하여 접근

클래스와 객체 생성

```
// 클래스 정의하기
class Person {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }

  sayHello() {
    console.log(`안녕하세요, ${this.name}입니다.`);
  }
}

// 객체 생성하기
const person1 = new Person("Jim", 30);
console.log(person1.name); // Jim
person1.sayHello(); // 안녕하세요, Jim입니다.
```

연습하기

상속과 다형성

자식 클래스가 부모 클래스의 속성과 메소드를 물려받아 사용하는 개념

부모 클래스의 메소드를 오버라이드하여 **동일한 메소드를 자식 클래스에서 다른 동작**을 수행하도록 하거나, 자식 클래스에서 **새로운 메소드를 추가**하여 다형성을 구현할 수도 있음

Vehicle

- Bikes
  - 
  - 
- Cars
  - 
  - 
- Buses
  - 
- Trucks
  - 

Animal

Inheritance

Dog

Cat

상속과 다형성

```
// 부모 클래스 정의하기
class Animal {
  constructor(name) {
    this.name = name;
  }

  speak() {
    console.log(`${this.name}이(가) 소리를 냅니다.`);
  }
}
```

```
// 자식 클래스 정의하기
class Dog extends Animal {
  constructor(name) {
    super(name);
  }

  speak() {
    console.log(`${this.name}이(가) 멍멍 짖습니다.`);
  }

  fetch() {
    console.log(`${this.name}이(가) 공을 가져옵니다.`);
  }
}
```

연습하기

모듈화와 패키지 매니저

모듈화란?

모듈화는 **코드를 기능별로 나누어 작성**하고, 필요한 부분만 가져와서 사용하는 방식. 모듈화를 통해 코드의 가독성과 재사용성을 높일 수 있음.

ES6 모듈

ES6 모듈은 모듈화를 위한 표준 방식으로, **파일 단위로 코드를 분리**하며, export와 import 구문을 사용하여 모듈을 가져오거나 내보낼 수 있음.

모듈화와 패키지 매니저

math.mjs

```
export function add(x, y) {
  return x + y;
}

export function subtract(x, y) {
  return x - y;
}
```

app.mjs

```
// main.mjs 모듈에서 math 모듈 가져오기
import * as math from "./math.mjs";

console.log(math.add(1, 2)); // 3
console.log(math.subtract(5, 3)); // 2
```

연습하기

NPM

NPM = Node Package Manager

NPM을 사용하면 **다른 개발자가 만든 Node.js 패키지를 쉽게 설치하여 사용**할 수 있음

```
npm install 패키지이름
```

AJAX와 HTTP 요청

AJAX란?

AJAX(Asynchronous JavaScript and XML)는 **비동기적으로 서버와 브라우저가 데이터를 주고받을 수 있는 기술**로 서버에서 데이터를 가져와서 동적으로 화면을 업데이트하는 방법

HTTP 요청과 응답

HTTP(Hypertext Transfer Protocol)

서버와 클라이언트 간에 데이터를 주고받는 프로토콜

HTTP 요청(**Request**)은 클라이언트에서 서버로 데이터를 보내는 것을 의미하며, HTTP 응답(**Response**)은 서버에서 클라이언트로 데이터를 보내는 것을 의미.

HTTP 메소드

HTTP 메소드는 클라이언트가 서버에게 요청하는 작업의 종류

GET: 서버로부터 리소스(데이터)를 가져오는 데 사용

POST: 서버에 새로운 리소스를 만드는데 사용

PUT: 서버에 있는 데이터를 변경하는데 사용

DELETE: 서버에서 리소스를 삭제하는 데 사용

HTTP 상태 코드

서버가 클라이언트에게 보내는 응답의 상태

- 1xx: 정보성 응답
- 2xx: 성공적인 응답

  200: 성공적인 GET 요청

  201: 성공적인 POST 요청

  204: 요청 성공, 응답에 내용 없음
- 3xx: 리디렉션

  301: 리소스가 새 위치로 이동됨

  302: 리소스가 일시적으로 새 위치로 이동됨

  304: 클라이언트 측의 캐시가 최신 상태임
- 4xx: 클라이언트 에러

  400: 잘못된 요청

  401: 인증 실패

  403: 권한 부족

  404: 찾을 수 없는 리소스
- 5xx: 서버 에러

  500: 서버 내부 오류

  502: 게이트웨이 오류

  503: 서비스 이용 불가능

Fetch API

Fetch API는 브라우저에서 제공하는 HTTP 요청을 처리하는 API

fetch API를 사용하면 AJAX 요청을 쉽게 처리할 수 있음. fetch 함수를 호출하고, then 메소드를 사용하여 응답 데이터를 처리.

Fetch API

```
let postData;

fetch("https://jsonplaceholder.typicode.com/todos/1")
  .then((response) => response.json())
  .then((data) => {
    console.log(data);
    postData = data;
  });
```

Fetch API

```
const postURL = "https://jsonplaceholder.typicode.com/todos";

fetch(postURL, {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  body: JSON.stringify(postData),
})
  .then((response) => response.json())
  .then((data) => console.log(data));
```

연습하기

TODO 앱 만들기

Todo App

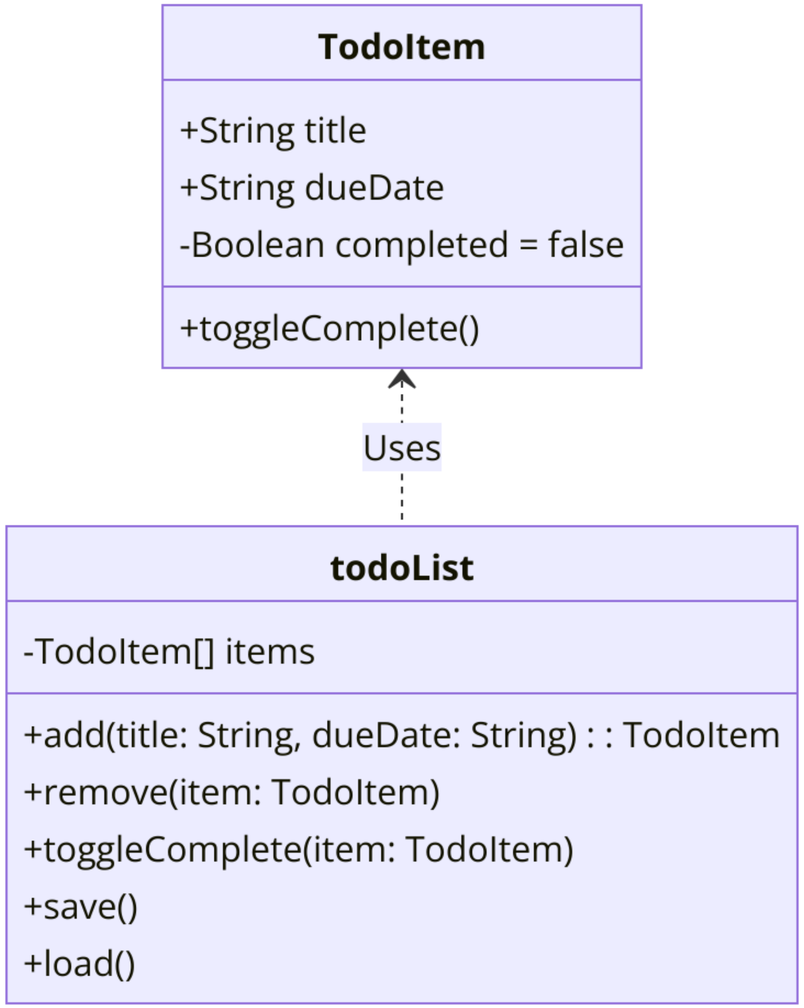
https://github.com/SNU-WP/javascript/

Todo App

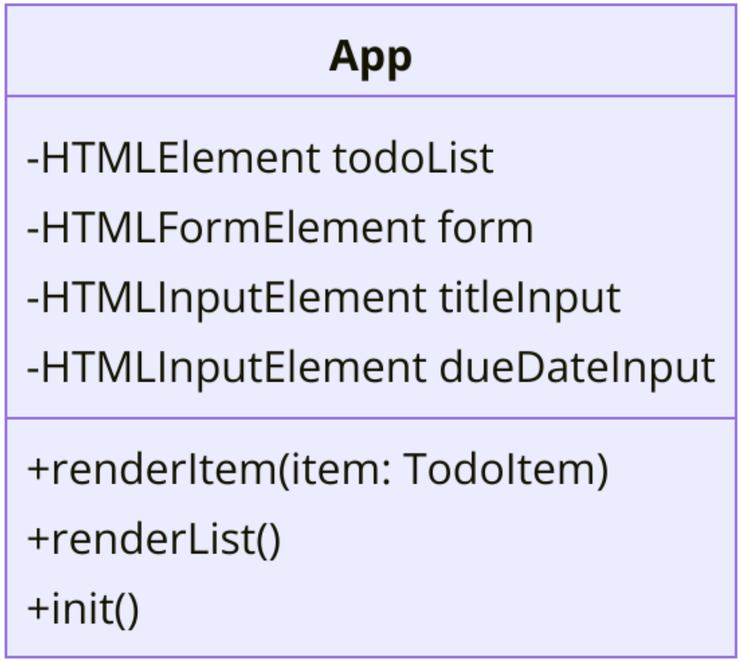
- index.html
- style.css
- item.js
  - 할 일의 항목 데이터를 처리하는 클래스
- todo.js
  - 할 일의 항목 데이터를 관리하는 모듈(객체)
- app.js
  - Todo 애플리케이션 UI 모듈

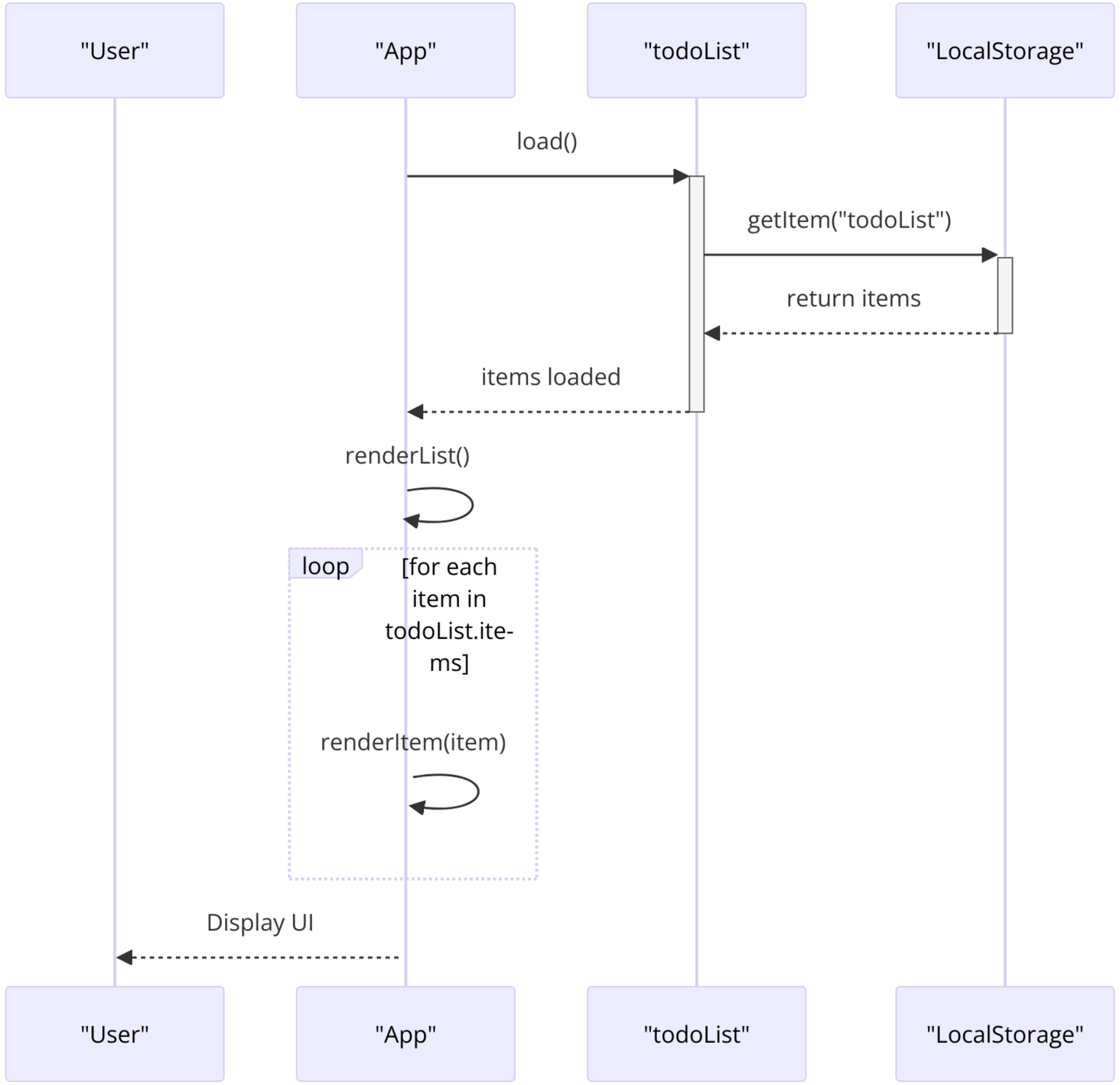
item.js, todo.js

```
items[
  TodoItem,
  TodoItem,
  ...
]
```

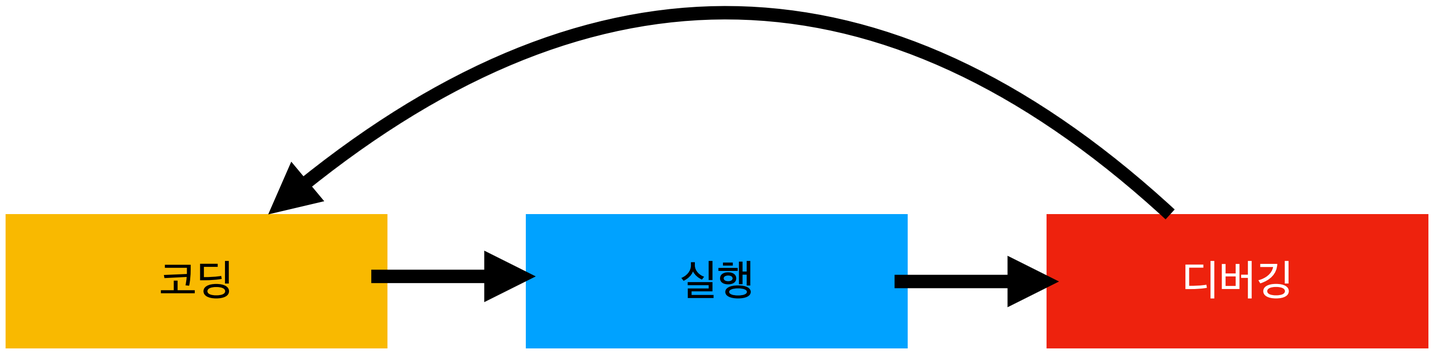


app.js


App
-HTMLElement todoList
-HTMLFormElement form
-HTMLInputElement titleInput
-HTMLInputElement dueDateInput
+renderItem(item: TodoItem)
+renderList()
+init()


"User"
"App"
"todoList"
"LocalStorage"
load()
getItem("todoList")
return items
items loaded
renderList()
loop
[for each item in todoList.items]
renderItem(item)
Display UI
"User"
"App"
"todoList"
"LocalStorage"

연습하기



소프트웨어 개발

프로그래밍 = 점진적 문제 해결

연습하기

1. todo 복사+붙여넣기로 실행해보기
2. 망가뜨리기
3. 어떻게 망가졌는지 확인하기
4. 고쳐보기
5. todo 폴더 아래 각 파일에 대해 2~4 세 번 반복하기

https://github.com/SNU-WP/javascript

연습하기 - 과제

1. AI와 함께 Todo 웹 앱 만들기
2. 3~4번 이상 AI와 함께 혹은 스스로 수정해보기
3. GitHub에 올리고 아래 Issue로 제출

https://github.com/SNU-WP/basecamp/issues/4