48장 모듈

48.1 모듈의 일반적 의미

- 모듈이란? — 애플리케이션을 구성하는 개별적 요소로 재사용 가능한 코드 조각
- 일반적으로 기능을 기준으로 파일 단위로 분리
- 모듈은 자신만의 파일 스코프(모듈 스코프)를 가질 수 있어야 한다. — 개별적 존재로서 애플리케이션과 분리되어 존재
- 모듈은 공개가 필요한 자산에 한정하여 명시적으로 선택적 공개가 가능 — 이를 export라고 함
- 모듈 사용자는 모듈이 공개한 자산 중 일부 또는 전체를 선택해 자신의 스코프 내로 불러들여 재사용할 수 있다. — 이를 import라고 함
- 모듈의 재사용성으로 인해 개발 효율성, 유지보수성을 높인다.

48.2 자바스크립트와 모듈

- ES6 전에는 모듈 시스템을 지원하지 않았다. — Script 태그를 통해 외부 자바스크립트 파일을 로드할 수는 있지만
  - 독립적 파일 스코프 지원 X
  - 전역을 공유하여 전역 변수가 중복되는 등의 문제 발생
- Script태그로만 모듈을 구성할 때 전역 변수 충돌등 많은 한계에 부딪힌다.
- ES6 이후 자바스크립트의 모듈 시스템은 크게 CommonJS, AMD 두 가지로 나뉘게 됐다.
  - Js 런타임 환경인 Node.js는 모듈 시스템의 사실상 표준인 CommonJS를 채택
  - Node.js는 ECMAScript 표준 사양은 아니지만 모듈 시스템을 지원

48.3 ES6 모듈(ESM)

0. 머릿말

- ES6는 클라이언트 사이드 js에서도 동작하는 모듈 기능을 추가했다.
  - import
  - export
- ES6 모듈을 ESM이라고 부른다.
- script태그에 type="module" 어트리뷰트 추가하면 로드된 js 파일은 모듈로서 동작 — 파일 확장자는 mjs 확장자 사용 권장
- ESM에는 클래스와 마찬가지로 기본적으로 strict mode가 적용

1. 모듈 스코프

- 모듈 내 변수는 전역 변수가 아니면서 window 객체의 프로퍼티도 아니다.
- 어트리뷰트에 tyep="module"을 넣으면 — 모듈 스코프가 다르기에 모듈 외부에서 참조 불가능하다.

2. export 키워드

- 모듈 내부에서 선언한 식별자를 외부에 공개하여 재사용할 수 있도록 하는 명령어
- 선언문 앞에 사용한다. — 변수, 함수, 클래스 등 모든 식별자
- 한 번에 객체로 묶어서 export 가능하다.

- 예제

3. import 키워드

- 다른 모듈의 export한 식별자를 내부로 로드하려면 import 명령어를 사용한다.
- ESM의 경우 파일 확장자 생략불가
- App의 진입점이 아니라면 Script 태그로 로드하지 않아도 된다.
- as 키워드 사용해서 이름 지정 가능
- 하나의 값만 export 한다면 export default 사용 할 수 있다.
  - Default 키워드를 사용하는 경우 기본적으로 이름 없이 하나의 값을 export
  - default키워드를 쓰는 경우 var, let, const 키워드는 사용 불가
  - 대신 import 할 때 { } 없이 임의의 이름으로 import 한다.

- 예제