Chap48 모듈

▼ 모듈의 일반적 의미

모듈 : 애플리케이션을 구성하는 개별적 요소로서 재사용 가능한 코드 조각을 말한다. 모듈이 성립하려면 자신의만의 파일 스코프(모듈 스코프)를 가질 수 있어야한다.

자신만의 파일 스코프를 갖는 모듈의 자산은 기본적으로 비공개 상태이지만 모듈은 애플리케이션이나 다른 모듈에 의해 재사용되어야 의미가 있다. 따라서 **공개가 필요한 자산에 한정하여 명시적으로 선택적 공개가 가능한다**. 이를 **export**라고 한다.

모듈 사용자는 모듈이 공개한 자산 중 일부 또는 전체를 선택해 자신의 스코프 내로 불러들여 재사용할 수 있다. 이를 **import**라 한다.

→ 이러한 모듈은 재사용성이 좋아서 효율성과 유지보수성을 높일 수 있다.

▼ 자바스크립트와 모듈

자바스크립트는 모듈 사스템을 지원하지 않았다. 즉 파일 스코프와 import, export를 지원하지 않았다.

하지만 CommonsJS와 AMD라는 것을 이용해 브라우저 환경에서 모듈을 사용하기 위해서는 CommonsJS와 AMD를 구현한 모듈 로더 라이브러리를 사용하면 가능했다.

따라서 자바스크립트 런타임 환경이 Node.js는 CommonsJS를 채택해 사용하고 있다.

Node.js는 ECMAScript 표준 사양은 아니지만 모듈시스템으 지원한다. 따라서 Node.js환경에서는 파일별로 독립적인 파일 스코프를 갖는다.

정리하자면 자바스크립트 자체는 모듈 시스템을 지원하지 않지만 자바스크립트 런타임 환경인 Node.js에서 CommonsJS를 구현한 모듈 로더 라이브러리를 사용하여 Node.js환경에서는 모듈 시스에 사용이 가능한다.

▼ ES6 모듈(ESM)

ES6에서는 클라이언트 사이드 자바스크립트에서도 동작하는 모듈 기능을 추가했다.

사용법 : script태그에 type=”module”어트리뷰트 추가하기

▼ 모듈 스코프

ESM은 독자적인 모듈 스코프를 갖는다. ESM이 아닌 일반적인 자바스크립트 파일은 독자적인 모듈 스코프 갖지 않는다.(위에서 말함)

▼ export키워드

선언문 앞에 위치

모듈 내부에서 선언한 식별자를 외부 공개하여 다른 모듈이 재사용할 수 있게 한다.

▼ import

다른 모듈에서 공개한 식별자를 자신의 모듈 스코프 내부로 로드할 때 사용