**Chapter 07. 익스프레스 프로젝트를 모듈화 하기**

* 모듈화 방법 살펴보기
* [user1.js]

exports 객체 속성으로 함수와 객체를 각각 추가한다. exports.getUser = function( ){ … }  
exports.group = {id: …, name: ….}

[module\_test1.js]

require로 user1 변수에 user1.js를 불러오고, return user1.getUser( ).name, user1.group.name을 출력할 수 있다.

**즉, exports에 속성으로서 함수와 객체를 할당하고, 이를 다른 파일에서 require로 참조하여 해당 클래스를 동일하게 사용할 수 있다.**

* [user2.js]

여기서는 동일한 객체와 함수를 exports의 속성이 아닌, 하나의 객체로서 할당한다.  
exports = { getUser : function…., group: { …} }

[module\_test2.js]

여기서 user2.js를 불러와 동일하게 출력하려 해도 에러가 발생한다.   
**🡪 이유는, exports에서 속성으로 객체와 함수를 할당하는 것은 가능하지만, user2.js와 같이 exports에 객체를 할당하게 되면 exports는 하나의 JS 단순 변수가 되어버린다. 이 경우에는 exports를 참조할 수 없게 된다.**

* + - 그래서 객체 안에 속성을 선언한 후, 해당 객체를 다른 파일에서 참조할 때는 **module.exports**를 사용해야 한다.
* [user3.js]

var user = { getUser : function( ){ …} , group : {….} } 처럼 객체를 선언하고, 마지막에 **module.exports = user;** 라고 객체를 할당함. 이후 module\_test3.js에서 require로 user3를 불러온 후 정상적으로 showUser( ) 에서 user.getUser().name 호출 가능

* module.exports에 함수만 할당하기

module.exports=function( ){….} 로도 직접 함수를 할당 가능하다. 이 경우, 불러올 때 **return user( ).name** 와 같이 익명함수로 반환되는 형태로 사용이 가능함.

* module.exports와 exports를 동시에 사용할 경우, module.exports를 우선적으로 사용하며 exports를 무시한다.
* 모듈 분리 시 전형적인 코드 패턴

1. 함수를 할당하는 패턴

exports.printUser = function( ) { … } 로 정의한 후, 다른 파일에서 require(‘./user7).printUser; 라고 모듈을 불러왔을 경우, printUser( ); 로 함수를 실행할 수 있다. 이 케이스는 변수명 printUser에 ( )를 붙여준 케이스다.

1. 인스턴스 객체를 할당하는 경우

// 생성자

function User(id, name){

this.id = id;

this.name = name;

}

User.prototype.getUser = function(){

return {id: this.id, name: this.name};

}

User.prototype.group = {id: 'group1', name: '친구'};

User.prototype.printUser = function(){

console.log('user이름 : %s, group이름: %s', this.name, this.group.name);

}

module.exports = new User('test01', '아이즈원띵');

var user = require('./user8');

user.printUser();

* + - * exports.abcd = new User( … ) 처럼 속성으로 추가 후 로드도 가능

1. 프로토타입 객체를 할당하는 경우