KB금융그룹 | 국민의 평생 금융파트너

2024년 상반기 K-디지털 트레이닝

자바스크립트 기초 문법과 모듈

[KB] IT's Your Life

KB 국민은행

함수의 정의

```
function 함수명(인수, ...) {
   // 함수 로직

   return 반환값;
}
```

chapter02/sec01/func01.js

```
function getTriangle(base, height) {
  return base * height / 2;
}

console.log('삼각형의 면적:' + getTriangle(5, 2));
```

```
삼각형의 면적:5
```

1 화살표 함수

함수 리터럴 표현으로 정의

- 익명함수
- 변수에 배정해서 사용

chapter02/sec01/func02.js

```
var getTriangle = function(base, height) {
  return base * height / 2;
};

console.log('삼각형의 면적:' + getTriangle(5, 2));
```

```
삼각형의 면적:5
```

화살표 함수

- 다른 언어에서는 람다 함수라고 함

```
(인수, ...) => { 함수 본체 }
```

- 인수가 1개인 경우 괄호 생략 가능
  - 인수 => { 함수 본체 }
- 함수 본체가 1줄인 경우 { } 생략 가능, return 키워드 생략 가능

```
let getTriangle = (base, height) => {
  return base * height / 2;
};

console.log('삼각형의 면적:' + getTriangle(5, 2));
```

```
let getTriangle = (base, height) => base * height / 2;

let getCircle = radius => radius * radius * Math.PI;
```

chapter02/sec01/func03.js

```
let getTriangle = (base, height) => (base * height) / 2;

let getCircle = (radius) => radius * radius * Math.PI;

console.log('삼각형의 면적:' + getTriangle(5, 2));
console.log('원의 면적:' + getCircle(5));
```

```
삼각형의 면적:5
원의 면적:78.53981633974483
```

동기 처리와 비동기 처리

동기 처리 방식과 비동기 처리 방식

chapter02/sec02/sync.js

```
function displayA() {
  console.log('A');
}

function displayB() {
  console.log('B');
}

function displayC() {
  console.log('C');
}

displayA();
displayB();
displayC();
```

```
A
B
C
```

chapter02/sec02/async-1.js

```
function displayA() {
  console.log('A');
}

function displayB() {
  setTimeout(() => {
    console.log('B');
  }, 2000);
}

function displayC() {
  console.log('C');
}

displayA();
displayB();
displayC();
```

```
A
C
B
```

chapter02/sec02/async-2.js

```
function displayA() {
  console.log('A');
}

function displayB(callback) {
  setTimeout(() => {
    console.log('B');
    callback();
  }, 2000);
}

function displayC() {
  console.log('C');
}

displayA();
displayB(displayC);
```

```
A
B
C
```

자바스크립트의 비동기 처리 방법

방법	버전	기능
콜백 함수	기존부터 사용	함수 안에 또 다른 함수를 매개변수로 넘겨서 실행 순서를 제어합니다. 콜백 함수가 많아지면 가독성이 떨어질 수 있습니다.
프라미스	에크마스크립트 2015부터 도입	프라미스 객체와 콜백 함수를 사용해서 실행 순서를 제어합니다.
async/await	에크마스크립트 2017부터 도입	async와 await 예약어를 사용해서 실행 순서를 제어합니다.

콜백 함수

- 함수 이름을 콜백으로 사용하기

```
const order = (coffee) => {
   console.log(`${coffee} 주문 접수`);
   // 시간이 3초 걸리는 작업
}

const display = (result) => {
  console.log(`${result} 완료!`);
}
```

```
function order(coffee) {
      // 커피 주문
      // 3초 기다린 후 완료 표시
}
function display(result) {
      // 커피 완료 표시
}
```

예상하는 프로그램 흐름

콜백 함수

- 함수 이름을 콜백으로 사용하기

```
function order(coffee, callback) {
    // 커피 주문
    // 3초 기다린 후 콜백 실행
}
function display(result) {
    // 커피 완료 표시
}

order("아메리카노", display)
```

display 함수를 order 함수의 매개변수로 넘기기

chapter02/sec02/callback-1.js

```
const order = (coffee, callback) => {
  console.log(`${coffee} 주문 접수`);
  setTimeout(() => {
    callback(coffee);
  }, 3000);
};

const display = (result) => {
  console.log(`${result} 완료!`);
};

order('아메리카노', display);
```

```
아메리카노 주문 접수
아메리카노 완료!
```

익명으로 콜백 함수 작성하기

```
function displayLetter() {
  console.log("A");
  setTimeout( () => {
    console.log("B");
    setTimeout( () => {
      console.log("C");
      setTimeout( () => {
        console.log("D");
        setTimeout( () => {
          console.log("stop!");
        }, 1000);
      }, 1000);
    },1000);
  }, 1000);
}

displayLetter();
```

프라미스 Promise

- ES2015(ES6)에서 도입
- Callback Hell을 피할 수 있는 방법

- Promise 객체를 생성할 때 비동기 작업 함수를 전달
  - 비동기 작업함수의 매개변수
    - resolve 함수: 작업 성공시 호출할 함수
    - reject 함수: 작업 실패시 호출할 함수

  - Promise 객체의 메서드
    - then(result): // 작업성공시 resolve 함수의 매개변수가 전달
    - catch(err): // 작업 실패시 reject 함수의 매개변수가 전달

프라미스 Promise

```
let likePizza = true;
const pizza = new Promise((resolve, reject) => {
  if(likePizza)
    resolve('피자를 주문합니다.');
  else
    reject('피자를 주문하지 않습니다.');
});
```

```
pizza
 .then(result => console.log(result))
 .catch(err => console.log(err));
```

chapter02/sec02/async-2.js

```
let likePizza = true;
const pizza = new Promise((resolve, reject) => {
  if (likePizza)
    resolve('피자를 주문합니다.');
  else
    reject('피자를 주문하지 않습니다.');
});

pizza
  .then(result => console.log(result))
  .catch(err => console.log(err));
// 피자를 주문합니다.
```

피자를 주문합니다.

async/await

- prmise의 then을 여러 번 사용하는 경우 복잡해짐
- 비동기 처리를 동기 스타일로 표현할 수 있는 기법

- 주의! async가 선언된 함수 내에서만 await 사용 가능

```
async function() {
...
await 함수
}
```

chapter02/sec02/withoutAwait.js

```
fetch('https://jsonplaceholder.typicode.com/users')
  .then((response) => response.json())
  .then((data) => console.log(data))
  .catch((err) => console.log(err));
```

```
[
  {
    id: 1,
    name: 'Leanne Graham',
    username: 'Bret',
    email: 'Sincere@april.biz',
    address: {
      street: 'Kulas Light',
      suite: 'Apt. 556',
      city: 'Gwenborough',
      zipcode: '92998-3874',
      geo: [Object]
    },
    ...
]
```

chapter02/sec02/await.js

```
async function init() {
  try {
    const response = await fetch('https://jsonplaceholder.typicode.com/users');
    const users = await response.json();
    console.log(users);
  } catch (err) {
    console.error(err);
  }
}

init();
```

```
[
  {
    id: 1,
    name: 'Leanne Graham',
    username: 'Bret',
    email: 'Sincere@april.biz',
    address: {
      street: 'Kulas Light',
      suite: 'Apt. 556',
      city: 'Gwenborough',
      zipcode: '92998-3874',
      geo: [Object]
    },
    ...
]
```

프로그래밍에서 가장 기본적인 개념, 모듈

- 모듈 module
  - 기능별로 만들어 놓은 함수
  - 파일 형태로 저장하고 필요할 때마다 가져와서 사용

CommonsJS 모듈 시스템과 ES 모듈 시스템

- 노드가 출시될 당시 모듈 운영에 대한 표준이 없었음
- CommonsJS
  - require() 함수를 통해 모듈을 사용

- ES 모듈 시스템
  - 자바스크립트 표준 모듈 시스템
  - 노드 13.2 이후 버전부터 지원

chapter02/sec03/greeting.js

```
const user = '홍길동';

// 인사하는 함수
const hello = (name) => {
  console.log(`${name}님, 안녕하세요?`);
};

hello(user);
```

```
홍길동님, 안녕하세요?
```

chapter02/sec03/user.js

```
const user = '홍길동';
```

chapter02/sec03/hello.js

```
const hello = (name) => {
  console.log(`${name}님, 안녕하세요?`);
};
```

모듈 내보내기 – module.exports

```
module.exports = 외부로 내보낼 함수 또는 변수
```

chapter02/sec03/user.js

```
const user = '홍길동';

module.exports = user; // user 변수 내보내기
```

chapter02/sec03/hello.js

```
const hello = (name) => {
  console.log(`${name}님, 안녕하세요?`);
};

module.exports = hello; // hello 함수 내보내기
```

```
exports.hello = (name) => {
  console.log(`${name}님, 안녕하세요?`);
};
```

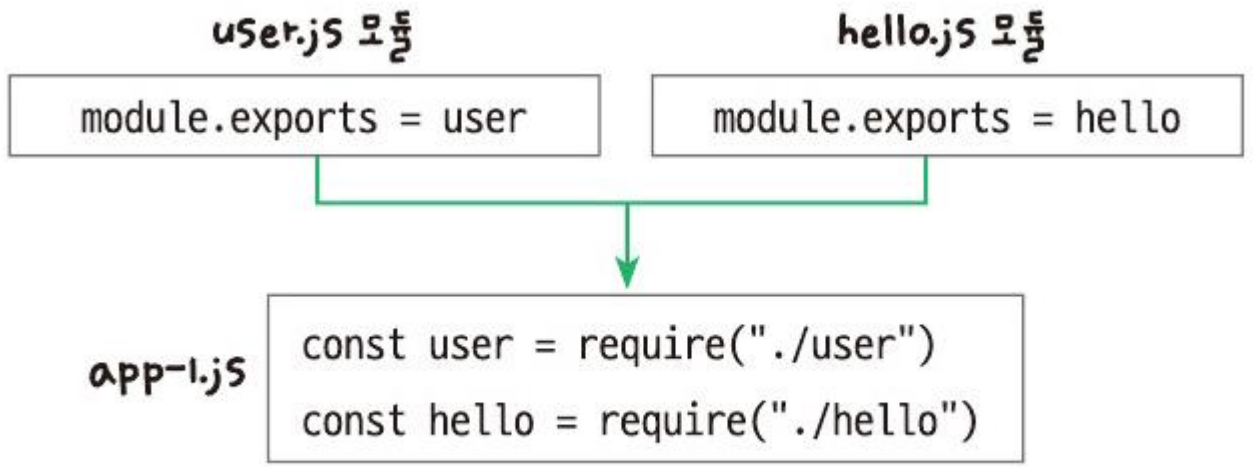
모듈 가져오기 – require 함수

```
require(모듈 파일 경로)
```

chapter02/sec03/app-1.js

```
const user = require('./user'); // user.js에서 user 가져오기
const hello = require('./hello'); // hello.js에서 hello 가져오기

console.log(user);
console.log(hello);

hello(user); // 모듈에서 가져온 user 변수와 hello 함수 사용하기
```

```
홍길동
[Function: hello]
홍길동님, 안녕하세요?
```



모듈을 활용한 프로그램

두개 이상의 변수 내보내기 및 가져오기

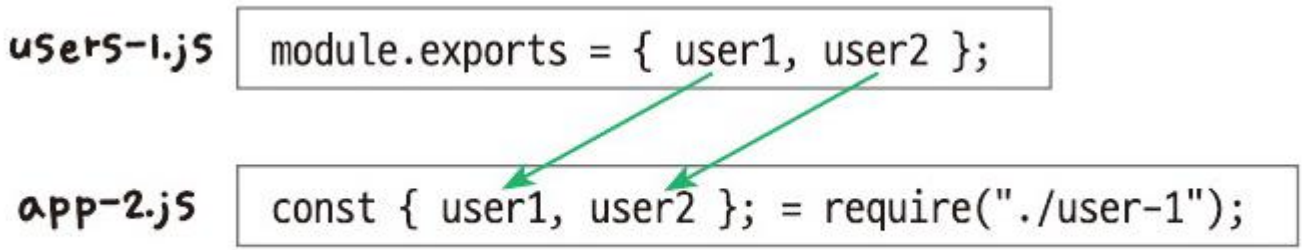
chapter02/sec03/users-1.js

```
const user1 = 'Kim';
const user2 = 'Lee';
const user3 = 'Choi';

module.exports = { user1, user2 };
```

chapter02/sec03/app-2.js

```
const { user1, user2 } = require('./users-1');
const hello = require('./hello');

hello(user1);
hello(user2);
```

```
Kim님, 안녕하세요?
Lee님, 안녕하세요?
```



변수 2개 내보내기

기능	모듈명	설명
파일 시스템	fs	파일이나 폴더에 접근할 수 있는 기능을 제공합니다. 예를 들어 파일 읽기/쓰기/삭제/이동/이름 변경이나 폴더 작업을 처리할 수 있습니다. 자세한 사용법은 03-2절에서 설명합니다.
HTTP	http	HTTP 서버를 만들고 요청을 처리하는 기능을 제공합니다. 익스프레스 같은 프레임워크 없이 HTTP 서버를 만드는 방법은 04-2절에서 설명합니다.
경로	path	파일 경로와 관련된 작업을 하는 기능을 제공합니다. 예를 들어 파일 경로를 지정하거나 상대 경로를 계산하는 작업을 할 수 있습니다. 자세한 사용법은 03-1절에서 설명합니다.
스트림	streams	데이터 스트림을 처리하는 기능을 제공합니다. 예를 들어 파일이나 네트워크와 같은 스트림에서 데이터를 읽거나 쓰는 작업을 할 수 있습니다. 스트림과 streams 모듈을 사용하는 방법은 03-5절에서 설명합니다.
암호화	crypto	암호화와 관련된 기능을 제공합니다. 해시 함수, 암호화 알고리즘, 암호화 및 복호화 등을 지원합니다.
운영체제	os	운영체제와 상호 작용하는 기능을 제공합니다. 예를 들어 운영체제의 정보를 알아내거나 시스템 리소스 정보를 확인할 수 있습니다.
유틸리티	util	다양한 유틸리티 함수를 제공합니다. 예를 들어 객체 상속, 비동기 처리를 프라미스로 변환하는 등의 작업을 할 수 있습니다.
이벤트	events	이벤트 기반 프로그래밍을 지원하는 기능을 제공합니다. 이벤트 생성, 등록, 처리 등을 할 수 있으며, 커스텀 이벤트를 사용하여 비동기 처리를 할 수 있습니다.

글로벌 모듈

- __dirname
  - 현재 모듈의 디렉토리 경로
- __filename
  - 현재 모듈의 파일 경로

chapter02/sec04/here.js

```
console.log(`현재 모듈이 있는 폴더 경로: ${__dirname}`);
console.log(`현재 모듈이 있는 파일 경로: ${__filename}`);
```

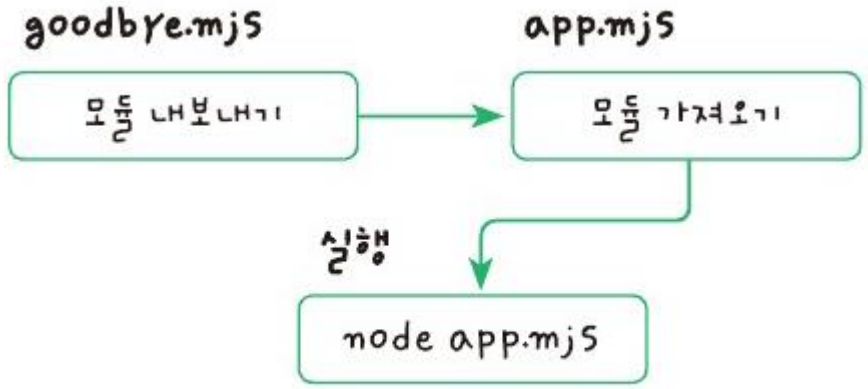
```
현재 모듈이 있는 폴더 경로: c:₩fullstack₩02_Node.js₩chapter02₩sec03
현재 모듈이 있는 파일 경로: c:₩fullstack₩02_Node.js₩chapter02₩sec03₩14_here.js
```

package.json에 설정하는 방법

```
{
  "name": "src",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "nodemon app.js"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "ansi-colors": "^4.1.3"
  },
  "type": "module"
}
```

파일 확장자를 .mjs로 지정

- CommonJS 모듈 시스템과 ES 모듈 시스템을 같이 사용할 때 사용
- package.json에 모듈 시스템 지정하면 안됨



ES 모듈 시스템에 사용하는 .mjs 확장자

ES 모듈 시스템에서 모듈 내보내기 – export, export default

- 변수나 함수 앞에 export를 붙임
  - 변수는 반드시 상수여야 함

```
export 대상
```

chapter02/sec05/goodbye-1.mjs

```
export const goodbye = (name) => {
  console.log(`${name} 님, 안녕히 가세요.`);
};
```

기본으로 내보내기 – export default

- 모듈에서 내보낼 대상이 하나 뿐일 때

```
export default 대상
```

chapter02/sec05/goodbye-2.mjs

```
const goodbye = (name) => {
  console.log(`${name} 님, 안녕히 가세요.`);
};

export default goodbye;
```

여러 개 내보내기

- 객체로 묶어서 내보냄

```
export {대상1, 대상2, ...}
```

chapter02/sec05/greeting-1.mjs

```
const hi = (name) => {
  console.log(`${name}님, 안녕하세요?`);
};

const goodbye = (name) => {
  console.log(`${name}님, 안녕히 가세요.`);
};

export { hi, goodbye };
```

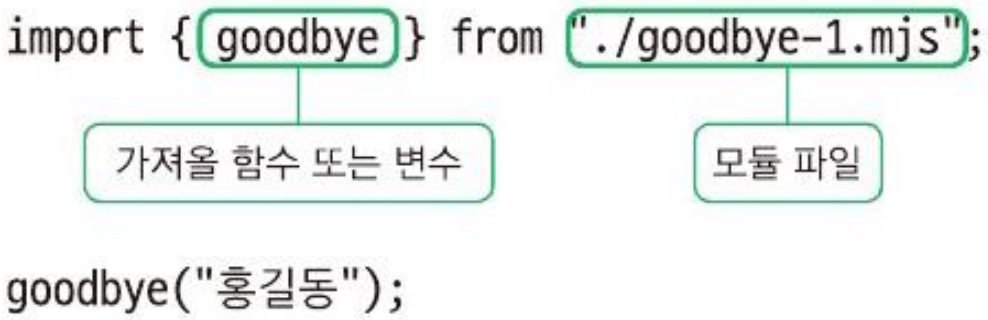
ES 모듈 시스템에서 모듈 가져오기 – import ~ from

```
import 변수명/함수명 from 모듈_파일
```

chapter02/sec05/app-5.mjs

```
import { goodbye } from './goodbye-1.mjs';

goodbye('홍길동');
```

```
홍길동 님, 안녕히 가세요.
```



chapter02/sec05/app-6.mjs

```
import { hi, goodbye } from './greeting-1.mjs';

hi('홍길동');
goodbye('홍길동');
```

```
홍길동님, 안녕하세요?
홍길동님, 안녕히 가세요.
```

- **import ~ as**
  - 가져오는 함수나 변수의 이름을 바꿔서 받을 수 있음

chapter02/sec05/app-7.mjs

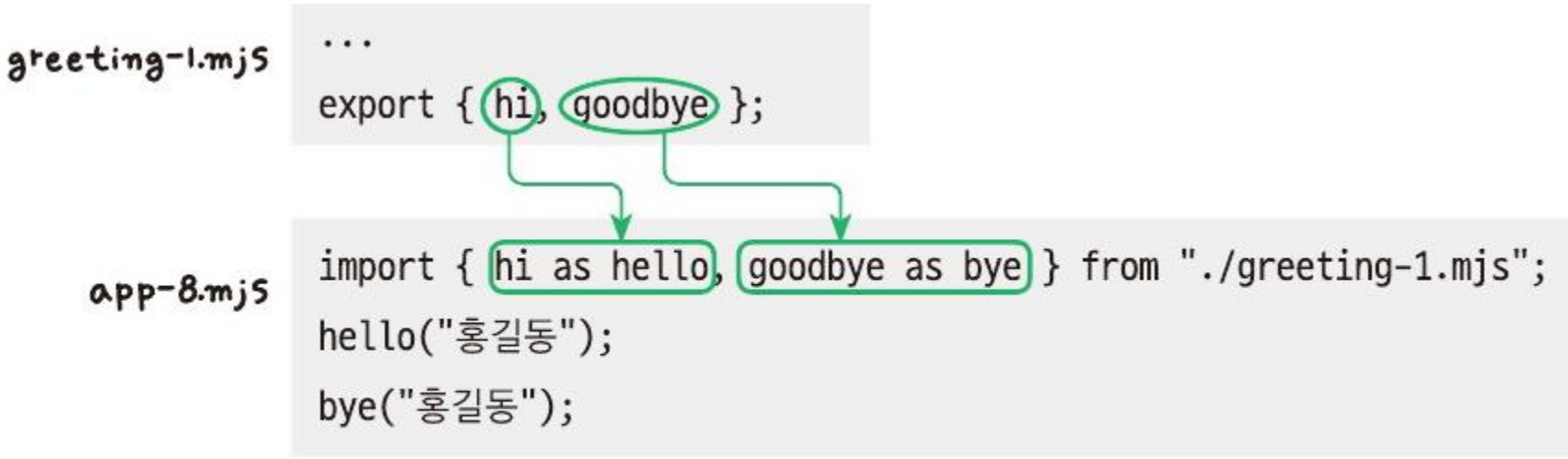
```
import { goodbye as bye } from './goodbye-1.mjs';

bye('홍길동');
```

```
홍길동님, 안녕히 가세요.
```

chapter02/sec05/app-8.mjs

```
import { hi as hello, goodbye as bye } from './greeting-1.mjs';

hello('홍길동');
bye('홍길동');
```

```
홍길동님, 안녕하세요?
홍길동님, 안녕히 가세요.
```



ES 모듈 시스템에서 둘 이상의 모듈 이름 바꾸기

import * as

- 모듈에서 가져와야 할 것이 너무 많은 경우 사용
- 해당 이름의 객체로 묶어 줌

chapter02/sec05/app-9.mjs

```
import * as say from './greeting-1.mjs'; // greeting.mjs에서 내보낸 함수들을 한꺼번에 say로 받기

say.hi('홍길동');
say.goodbye('홍길동');
```

```
홍길동님, 안녕하세요?
홍길동님, 안녕히 가세요.
```

- export default 가져오기

chapter02/sec05/greeting-2.mjs

```
const hi = (name) => {
  console.log(`${name}님, 안녕하세요?`);
};

const goodbye = (name) => {
  console.log(`${name}님, 안녕히 가세요.`);
};

export default { hi, goodbye };
```

chapter02/sec05/app-10.mjs

```
import say from './greeting-2.mjs';

say.hi('홍길동');
say.goodbye('홍길동');
```