Seoul woman up

스타트업 개발자와 함께 공부하는

Node.js

03. 패키지매니저 및 모듈

강의 내용은 강사가 별도로 명시하지 않는 한 비공개로 간주합니다. 녹음이나 사진 촬영를 허락하지 않으며 콘텐츠를 블로그, SNS 등에 게시하거나 공개적으로 공유하지 마세요.

콘텐츠 공유 가능 여부에 대해 궁금한 점이 있는 경우 강사에게 문의하시기 바랍니다.

목차

패키지 매니저

패키지 매니저

npm(node package manager)

- npm 은 자바스크립트 패키지 매니저임
- node.js 에서 사용할 수 있는 모듈을 패키지화 하여 모아둔 장소
- CLI(Command line interface) 제공

패키지 매니저

npm command

- **npm install <packagename>@<version>**
- **npm uninstall <packagename>@<version>**
- **npm update <packagename>@<version>**
- **npm view <packagename>@<version>**

패키지 매니저

npm version

- npm 버전 형식 : {major}.{minor}.{path}
- 2.1.0
  - 2 : 메이저 (큰 기능 추가)
  - 1 : 마이너 (작은 기능 추가)
  - 0 : 패치 (버그나 사소한 오류)

패키지 매니저

npm version

- 틸드(~): x.y.z 중 z 범위 내에서 버전 업데이트
- 캐럿(^): x.y.z 중 x 이하 하위호환성이 보장되는 범위 내에서 버전 업데이트

- ~1.2.3 `>= 1.2.3 < 1.3.0`

- ^1.2.3 `>= 1.2.3 < 2.0.0`

확인 문제

npm install

1. 파워쉘 또는 도스창에서 [C]-[nodejs]-[project]-[03] 로 이동합니다
2. npm 을 이용해 최신 버전의 express 를 설치해보세요
3. 설치된 express 를 지워보세요
4. 3.21.2 버전의 express를 설치해보세요
5. 설치된 express를 지워보세요.
6. 4.19.X 중에서 가장 최신 패치의 express를 설치해보세요

확인 문제

외부 모듈 'express'를 프로젝트에 설치하기 위한 명령어는 무엇인가요?

1. npm install express
2. node install express
3. install express
4. npm require express

새로운 Node.js 프로젝트를 초기화하기 위한 명령어는 무엇인가요?

1. npm init
2. npm start
3. npm install
4. node init

package.json

package.json

프로젝트 생성

- npm init 명령어 : 노드 프로젝트를 생성하는데 사용함
- package.json 은
  - 노드 프로젝트의 정보, 의존성, 스크립트 등을 포함하고 있음

[C]-[nodejs]-[project]-[03]

```
~/W/t/nodejs/day03 on main !6 ?9 npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (day03) test project
Sorry, name can only contain URL-friendly characters.
package name: (day03) test_project
version: (1.0.0) 1.0.2
description: 테스트 프로젝트입니다
entry point: (index.js) server.js
test command:
git repository:
keywords: 테스트, test
author: mhb8436
license: (ISC) MIT
About to write to /Users/mhb8436/Workspaces/tutorial/nodejs/day03/package.json:

{
  "name": "test_project",
  "version": "1.0.2",
  "description": "테스트 프로젝트입니다",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "테스트",
    "test"
  ],
  "author": "mhb8436",
  "license": "MIT"
}


Is this OK? (yes) yes
```

package.json

```
{
  "name": "day08",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "test": "jest",
    "dev": "nodemon server.js"
  },
  "keywords": [
    "user"
  ],
  "author": "mhb8436",
  "license": "ISC",
  "dependencies": {
    "bcryptjs": "^2.4.3",
    "express": "^4.19.2",
    "jsonwebtoken": "^9.0.2",
    "nodemon": "^3.1.3",
    "pg": "^8.12.0",
    "sequelize": "^6.37.3",
    "sequelize-cli": "^6.6.2",
    "sqlite3": "^5.1.7"
  },
  "devDependencies": {
    "@babel/plugin-transform-modules-commonjs": "^7.15.0",
    "@babel/preset-env": "^7.15.6",
    "@babel/preset-typescript": "^7.15.0",
    "jest": "^29.7.0",
    "sequelize-mock": "^0.10.2"
  },
  "jest": {
    "transform": {
      "^.+\\.jsx?$": "babel-jest"
    }
  }
}
```

파일 설명

- scripts
- dependencies
- devDependecies
- jest

확인 문제

package.json 에 포함되는 기본 필드는 무엇인가요?

1. author
2. dependencies
3. name
4. license

package.json 파일에서 dependencies 필드는 무엇을 정의하나요?

1. 프로젝트 이름
2. 프로젝트 버전
3. 프로젝트가 필요하는 외부 모듈
4. 프로젝트 시작 스크립트

모듈 만들고 사용하기

모듈 만들고 사용하기

모듈, 패키지, 라이브러리

- 모듈 : 특정 기능들이 구현되어 있는 코드가 적혀 있는 파일
- 패키지 : 이런 모듈을 상위 폴더에 넣어 패키징 한 것
- 라이브러리 : 이러한 모듈과 패키지들의 묶음

모듈 만들고 사용하기

모듈 내보내기 함수 , 모듈 불러오기

- 모듈을 내보낼 때는 module.exports 를 이용

```
module.exports = data
```

- require 함수는 노드에서 모듈을 불러올 때 사용하는 함수

```
const data = require(”../data”)
```

모듈 만들고 사용하기

```
1  const  add = (a,b) => a+b;
2
3  // for test
4  // console.log(module)
5
6  // 1
7  module.exports = add
8  console.log(module)
9
10 // 2
11 module.exports.add = add;
12 console.log(module)
```

- 모듈 내보내기
- 소스파일 : ch03_01.js
- module.exports 를 사용하면 특정 자바스크립트 내용을 모듈화 시켜 외부로 낼 수 있음

모듈 만들고 사용하기

```
1 // 1
2 const add = require('./ch02_01');
3 console.log(add(4,2));
4
5 // 2
6 const calc = require('./ch02_01');
7 console.log(calc.add(4,2));
8
9
```

- 모듈 불러오기
- 소스코드파일 : ch03_02.js
- require 함수 이용하여 모듈을 불러 올 수있음

확인 문제

Node.js 파일 작업을 위해 fs 모듈을 가져오는 올바른 방법은?

1. import fs from ‘fs’;
2. const fs = require(‘fs’);
3. require(‘fs’);
4. import ‘fs’;

현재 디렉토리에 있는 myModule.js 파일을 가져오기 위한 올바른 방법은 무엇인가요?

1. const myModule = require(‘./myModule’);
2. const myModule = require(‘myModule’);
3. import myModule from ‘./myModule’;
4. require(‘./myModule’);

파일 읽고 쓰기

파일 읽고 쓰기

fs 모듈 이용

- 파일을 읽을 때

```
const fs = require('fs')
fs.readFile(filename, 'utf-8', ()=>{})
```

- 파일을 쓸 때

```
const fs = require('fs')
fs.writeFile(filename, data, err=>{})
```

파일 읽고 쓰기

```
1 const fs = require('fs')
2
3 fs.readFile('hello.txt', 'utf8', (err,data)=> {
4     if(err) {
5         console.log('err', err);
6     }
7     console.log(data);
8 });
9
```

- 파일 읽기
- 소스코드 파일명 : ch03_03.js

파일 읽고 쓰기

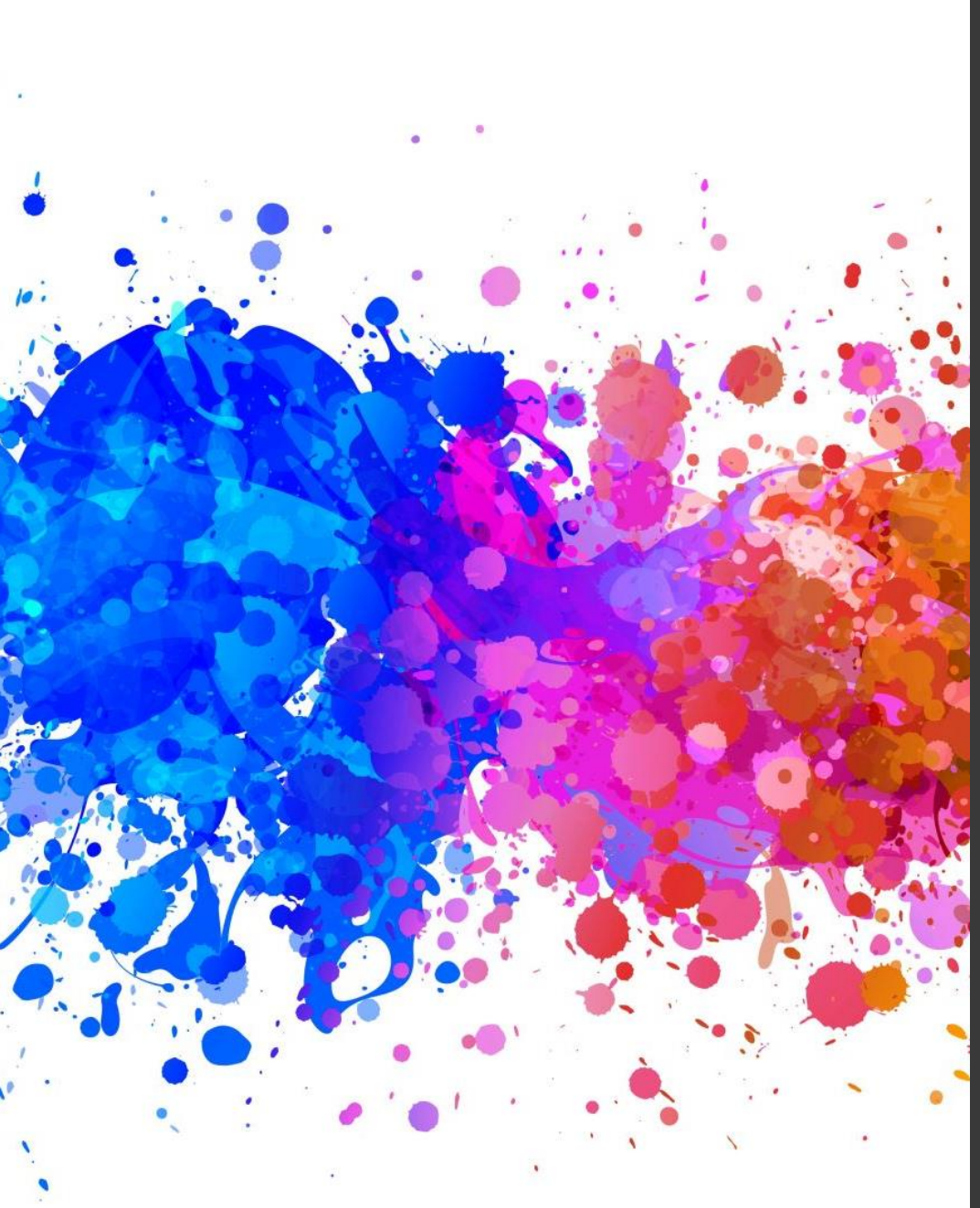
```
1 const fs = require('fs')
2
3 const content = 'this is content'
4
5 fs.writeFile('content.txt', content, err => {});
6
7 // fs.writeFileSync('content.txt', content);
8
```

- 파일 쓰기
- 소스코드 파일명 : ch03_04.js

파일 읽고 쓰기

UTF-8

- 초기 문자열 표(ASCII)에서는 알파벳과 숫자만 존재
- 한글 표현 불가능, 나라들 마다 문자열 표를 각각 만듬
- 한글은 ECU-KR
- 문자열 세트가 따로 있어서 문자가 깨지는 현상이 자주 발생
- 전 세계 문자가 통합된 문자열 세트를 만듬 → 유니코드 → UTF-8은 유니코드를 인코딩하는 방식



JSON 파일

JSON

JSON 소개

- JavaScript Object Notation
- 데이터를 저장하고 교환하기 위한 경량 데이터 교환 형식

```
<widget>
    <debug>on</debug>
    <window title="Sample Konfabulator Widget">
        <name>main_window</name>
        <width>500</width>
        <height>500</height>
    </window>
    <image src="Images/Sun.png" name="sun1">
        <hOffset>250</hOffset>
        <vOffset>250</vOffset>
        <alignment>center</alignment>
    </image>
    <text data="Click Here" size="36" style="bold">
        <name>text1</name>
        <hOffset>250</hOffset>
        <vOffset>100</vOffset>
        <alignment>center</alignment>
        <onMouseUp>
            sun1.opacity = (sun1.opacity / 100) * 90;
        </onMouseUp>
    </text>
</widget>
```

```
{"widget": {
    "debug": "on",
    "window": {
        "title": "Sample Konfabulator Widget",
        "name": "main_window",
        "width": 500,
        "height": 500
    },
    "image": {
        "src": "Images/Sun.png",
        "name": "sun1",
        "hOffset": 250,
        "vOffset": 250,
        "alignment": "center"
    },
    "text": {
        "data": "Click Here",
        "size": 36,
        "style": "bold",
        "name": "text1",
        "hOffset": 250,
        "vOffset": 100,
        "alignment": "center",
        "onMouseUp": "sun1.opacity = (sun1.opacity / 100) * 90;"
    }
}}
```

JSON

JSON 특징

- 간결함
- 호환성
- 유연성

JSON

JSON 문법

- 객체(Object)
- 배열(Array)
- 키-값 쌍(Key-Value Pair)

```
{
    "name": "Alice",
    "age": 25,
    "address": {
        "street": "123 Main St",
        "city": "Somewhere"
    },
    "phoneNumbers": [
        "123-456-7890",
        "987-654-3210"
    ]
}
```

JSON

JSON 사용 예

- 웹 애플리케이션에서 데이터 교환
- 구성 파일(Configuration)

```
// 서버에서 클라이언트로 전송되는 응답 예시
{
    "status": "success",
    "data": {
        "userId": 1,
        "username": "john_doe"
    }
}
```

```
{
    "version": "1.0.0",
    "settings": {
        "theme": "dark",
        "language": "en"
    }
}
```

JSON

JSON 구문 분석

- JSON → 자바스크립트 객체 : **JSON.parse()**
- 자바스크립트 객체 → JSON : **JSON.stringify()**

```
const jsonString = '{"name": "John", "age": 30}';
const jsonObject = JSON.parse(jsonString);
console.log(jsonObject.name); // "John"

const newJsonString = JSON.stringify(jsonObject);
console.log(newJsonString); // '{"name":"John","age":30}'
```

JSON

```
1  const fs = require('fs')
2
3  let result = []
4
5  for(i=1;i<11;i++){
6      result.push({'title': 'This is Title ' +i,
7          'content':'This is Content ' + i});
8  }
9
10 const data = {
11     'result': result
12 };
13
14 fs.writeFileSync('test.json',
15     JSON.stringify(data, null, 2), 'utf-8');
```

- JSON 파일 쓰기
- 소스코드파일명 : ch03_05.js
- 자바스크립트 객체를 JSON 파일로 저장

JSON

```
1  const fs = require('fs');
2
3  const result = fs.readFileSync('test.json', 'utf-8');
4
5  const data = JSON.parse(result);
6
7  data['result'].forEach(x=> {
8      console.log(x['title'], x['content']);
9  });
10
```

- JSON 파일 읽기
- 소스코드파일명: ch03_06.js

백문이 불여일타