ლაბორატორიული 6

1. **npm init, npm install, scripts**

**init**

პროექტის ინიციალიზაციისთვის საჭიროა კონსოლში შემდეგი ბრძანების შესრულება:

>npm init

მას შემდეგ რაც რამდენიმე კითხვას ვუპასუხებთ, პროექტის ფოლდერში შეიქმნება ფაილი packages.json დაახლოებით შემდეგი შიგთავსით:

{

"name": "lab7",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"test": "test"

},

"repository": {

"type": "git",

"url": "lab7"

},

"author": "gdznela",

"license": "MIT"

}

ამ ფაილში მოთავსბულია ყველა ინფორმაცია რაც ჭირდება npm-ს მუშაობისთვის.

**install**

რაიმე ახალი js მოდულის დასაყენებლად გამოიყენება ბრძანება npm install, რომელიც ასევე ავტომტურად აყენებს მოდულის მუშაობისთვის ყველა სხვა დანარჩენ საჭირო მოდულს (dependency-ებს). მაგალითად ajax-თან მუშაობისთვის axios-ის დასაყენებლად საჭიროა შემდეგი ბრძანების შესრულება:

>npm install axios --save

--save ოპციის გადაცემა საჭიროა რათა მოხდეს package.json-ში დაფიქსირება რომ ეს მოდული ესაჭიროება პროექტს, ასევე მისი ვერსია (ჩაემატება dependencies-ში). ბრძანების შესრულების შედეგად ასევე შეიქმნება ფოლდერი node\_modules სადაც ინახება ყველა დაყენებული მოდულის ფაილი.

როდესაც ვაკომიტებთ კოდს source control-ში უმეტეს შემთხვევაში ვაიგნორებთ node\_modules ფოლდერს(.gitignore). ამიტომ source control-დან ახალ აღებულ პროექტში საჭირო მოდულები არ არის დაყენებული სანამ არ გავუშვებთ ბრძანებას npm install პარამეტრების გარეშე. ბრძანების შესრულების შედეგად მოხდება packages.json წაკითხვა და ყველა აუცილებელი მოდულის დაყენება რომელიც მითითებულია მასში.

ხანდახან ზოგი მოდული საჭიროა მხოლოდ დეველოპმენტის დროს, მაგალითად js bundling ინსტრუმენტები როგორიცაა browserify. ასეთი მოდულების დაყენებისას მიეთითება ოპცია --save-dev. მაგალითად browserify-ს დასაყენებლად:

>npm install browserify --save-dev

შედეგად browserify დაემატება package.json-ში devDepenendcies-ში.

რაიმა მოდულის გლობალურად დასაყნეებლად (არა მარტო ლოკალურ ფოლდერში არამედ ნებისმიერ ადგილას რომ ხელმისაწვდომი იყოს კონსოლში) გამოიყენება -g ოპცია, მაგალითად:

npm install browserify -g

**scripts**

ხანდახან პროექტის დაბილდვისთვისან სხვა ამოცაების შესასრულებლად ხშირად გვიწევს კონსოლიდან რაღაც ბრძანების შესრულება და რამდენიმე პარამეტრის გადაცემა. ამ პროცესის გასამარტივებლად შესაძლებელია package.json-ში scripts-Ыш დავამატოთ ეს ბრძანებები თავისი პარამეტრებით და მივანიჭოთ სახელი. შემდგომ ბრძანების გამოძახებისთვის საკმარისი იქნება მხოლოდ სახელის მითთება. მაგალითად თუ scripts-ში არის შემდეგი:

"scripts": {

"build": "browserify --entry index.js --outfile bundle.js",

"buildw": "watchify index.js -o bundle.js -v"

},

browserify-ის ბილდის შესასრულებლად საკმარისი იქნება კონსოლში დავწეროთ:

>npm run build

1. **browserify**

npm მოდულების დაყენებისას მოდულები შესაძებელია იყოს სხვა და სხვა ფორმატში (commonjs, AMD ასევე არსებობს ES6 native). რადგან nodejs-გან გასხვავებით ბრაუზერებს ჯერ არ აქვთ მოდულების native მხარდაჭრა საჭიროა პროექტის დაბილდვა, ანუ ყველა საჭირო მოდულის კოდის გაერთინაბა ერთ ან რამდენიმე ფაილში რომელსაც bundle ეწოდება. ამისთვის არსებობს სხვადასხვა ინსტრუმენტი მაგალითად browserify, webpack. Webpack-ს შედარებით მეტი შესაძლებლობა აქვს, თუმცა მისი კონფიგურირება შედარებით რთულია. ამიტომ პირველ მაგალითებში გამოვიყენებთ browserify-ს.

Browserify-ის ბილდის გაშვების მაგალითი:

browserify --entry index.js --outfile bundle.js

პარამეტრები: entry - ჩვენი ძირითად js ფაილის სახელი, სადაც ხდება საჭირო მოდულების იმპორტი require-ს გამოყენებით. --outfile ბილდის შედეგად შესაქმნელი bundle ფაილის სახელი.

1. **require**