RESUMO DA AULA

18 Gerenciando pacotes com NPM

Nesta aula, vamos aprofundar nosso conhecimento sobre o gerenciamento de pacotes usando o NPM (Node Package Manager) e entender como essa ferramenta utilizada na pr tica para criar e manter projetos JavaScript organizados e eficientes. Inicializando um Projeto com o NPM Para come ar a usar o NPM em um projeto, precisamos inicializar o gerenciador de pacotes dentro da pasta do projeto. Esse processo cria um arquivo package.json, que serve como um ponto central de controle de pacotes e configura es do projeto. Para isso, vamos usar o comando: npm init Esse comando precisa ser executado no terminal dentro da pasta raiz do projeto. Ao rodar npm init, o NPM nos guiar por uma s rie de perguntas para configurar o projeto. Vamos ver cada uma delas: Nome do projeto: Define o nome nico do projeto. importante que ele n o contenha espa os, por isso geralmente usamos - ou _ para separar palavras. Exemplo: projeto_npm. Vers o do projeto: Geralmente, come amos com 1.0.0, indicando a primeira vers o. Futuras atualiza es incrementam esses n meros. Descri o: Uma breve descri o do que o projeto faz. Exemplo: Projeto de teste para aprender NPM no m dulo de JavaScript Avan ado. Entry point (ponto de entrada): Arquivo principal do projeto, como index.js. Esse o arquivo que o Node.js procurar para iniciar a aplica o. Test command: Comando usado para rodar testes automatizados. Podemos deixar vazio se n o vamos usar testes por enquanto. Git repository: URL do reposit rio Git, se o projeto estiver vinculado a um controle de vers o. Keywords: Palavras-chave para identificar o projeto (separadas por v rgulas). Author: Nome do autor do projeto (voc !). License: Tipo de licen a. Podemos usar a licen a padr o (ISC). Ap s confirmar as informa es, o NPM gera um arquivo package.json contendo todas essas configura es. Instalando Pacotes com NPM Depois de configurar o package.json, podemos come ar a instalar pacotes no projeto usando o comando: npm install <nome-do-pacote> Por exemplo, vamos instalar um pacote chamado kind-of, que ajuda a identificar o tipo de vari veis: npm install kind-of --save Esse comando faz duas coisas: Atualiza o package.json: Ele cria uma se o chamada

dependencies e adiciona o pacote kind-of como depend ncia. "dependencies": { "kind-of": "^6.0.3" } Cria a pasta node_modules: Dentro dessa pasta, o NPM armazena todos os pacotes e suas depend ncias. importante lembrar que essa pasta apenas para uso local e n o deve ser enviada para o controle de vers o. Al m disso, o NPM tamb m cria um arquivo package-lock.json, que cont m informa es detalhadas sobre a rvore de depend ncias do projeto. Esse arquivo essencial para garantir que todos os colaboradores do projeto tenham exatamente as mesmas vers es de pacotes instaladas. Utilizando Pacotes no Projeto Depois de instalar um pacote, podemos utiliz -lo no nosso projeto usando o require. Por exemplo: var kindOf = require('kind-of'); console.log(kindOf(true)); // Output: 'boolean' O m todo require carrega o pacote kind-of e permite usar suas funcionalidades no nosso c digo. Vale lembrar que esse m todo espec fico para o ambiente Node.js e n o funcionar diretamente no navegador. Gerenciando o node_modules no Controle de Vers o A pasta node_modules geralmente cont m centenas ou at milhares de arquivos e n o deve ser enviada para o reposit rio Git. Em vez disso, enviamos apenas o package.json e o package-lock.json. Assim, quem for trabalhar no projeto no futuro s precisa rodar o comando: npm install Esse comando verifica o package.json e o package-lock.json e baixa todas as depend ncias listadas, recriando a pasta node_modules no ambiente local. O que o package-lock.json? O package-lock.json um arquivo criado automaticamente pelo NPM sempre que instalamos ou removemos um pacote. Ele serve para garantir que o ambiente do projeto seja replicado exatamente igual em outras m quinas, especificando vers es exatas das depend ncias. Quando compartilhamos um projeto com outras pessoas, o package-lock.json ajuda a garantir que todos usem as mesmas vers es dos pacotes, evitando problemas de compatibilidade.
