08

Evitando Babel no escopo global

Muitos tutoriais da internet instalam Babel e outros módulos do Node.js globalmente por uma questão de brevidade, mas que não é uma boa prática.

Se você precisa da nova versão do Babel porque seu projeto A depende de um novo recurso, a atualização da instalação global será aplicada em todos os projetos. Ela pode funcionar perfeitamente em A, mas pode quebrar o projeto B que até então funcionava se algum BUG foi introduzido, um BUG que só afeta um recurso utilizado por B.

Sendo assim, instalamos Babel local ao projeto, contudo não é nada elegante a forma com que chamaremos manualmente o binário do babel em nosso terminal. Para contornar esse problema e ainda termos o babel instalado localmente para cada um dos nosso projetos, podemos criar um script em `package.json` que chamará o Babel para nós.

Qual das opções abaixo possui a chave `script` configurada corretamente para chamar `Babel` e compilar todos os nossos arquivos dentro da pasta `aluraframe/client/js/app-es6` resultando na pasta `aluraframe/client/js/app` :

Selecione uma alternativa

A

```
{

  // código omitido

  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "build": "babel js/app-es6 -d js/app "
  },

  // código omitido
}
```

B

```
{

  // código omitido

  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "build": "babel js/app-es6 js/app "
  },

  // código omitido
}
```

C

```
{
  // código omitido

  "scripts": {
```

```
    "test": "echo \"Error: no test specified\" && exit 1"
    "build": "babel js/app-es6 -d js/app "
  },

  // código omitido
}
```