28)how to publish a library to private enterprise npm repository

Now is the moment for publishing our newly developed module on npm. we are going to show 2 ways of doing it. We are going to show how to publish a library in the angular package format which is the recommended format for open source libraryies. On the other hand uf you are doing internal project and you are sure that your module will only be consumed by application that are using angular cli which is very common case, then in that case there is more simpler , more immediate way of publishing the libray to your enterprise repository. Lets look at second way first.

Now see code.

30)Material design Input box- Consolidation Exercise solution

Here we will use material icons. Go to this link-

<https://material.io/tools/icons/?style=baseline>

final html if we use material icons looks like this-

<i class="md-icon">search</i>

So we need to change the html of out component.

We did this-

In component html-

<i class="md-icon" [innerHtml]="icon"></i>

Here icon is passed from outside. Rest all things remained same.

Note we made a different component for material icons. One way was that we could use exisiting component and pass a additional parameter to tell which icon class to use (font awesome or material icon) to use. Instead of doing that we adopted a approach of creating small well defined components that have one specific functionality instead of creating one large component with api that then internally gets branched into effectively multiple components.

Now if we see code of both components, they are identical same except for small change in component as explained above. But in this case we have not created a common superclass and inherited from it etc. In this case and this is suggested practice in clean code book, for example when we are in doubt that if we should refactor things to make code more reusable, we should do that. may be if we had 3 or more different inputs and they have code, then we should thing of refactoring our code into a common super class or something. But in general in our code it is good idea to avoid extends keyword as much as possible. If you notice in public api of angular we never had to extend anything, we simply create plain classes and we add some decorator’s to them. But we are not encouraged to use object oriented inheritance. It’s not present in public api of angular and it’s a good idea not to use it our code , if it is not absolute necessary and preferring style approach based on composition

Inheritance in compoennts-

<https://coryrylan.com/blog/angular-component-inheritance-and-template-swapping>

in our case there were only couple of line that we re duplicated , so its fine to have coulple of extra line than to use extends.

So in summary while designing our components lets prefer to use composition over inheritance in general.