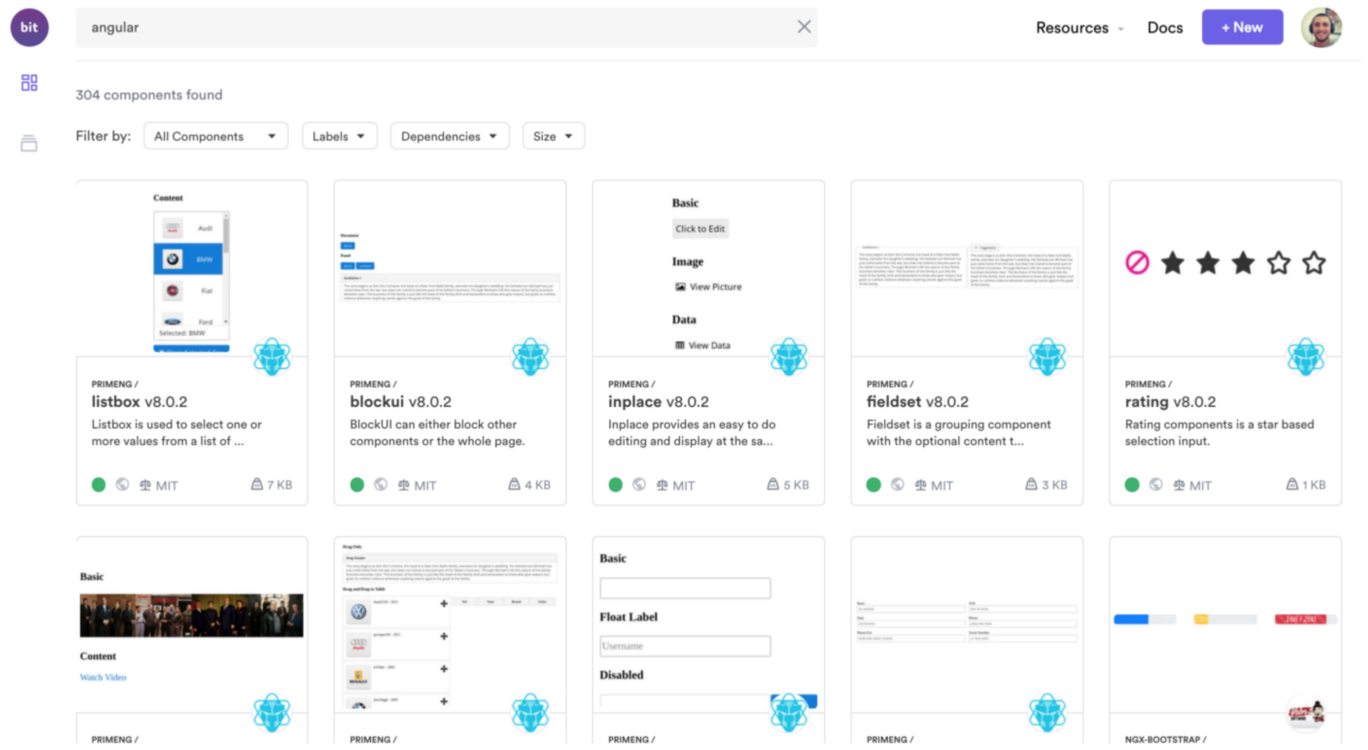
Bits and Pieces

How to Share Angular Components Between Projects and Apps

Easily share and collaborate on components across Angular projects, alone or as a team.

Jonathan Saring Follow
Aug 8, 2019 · 13 min read


bit
angular
Resources
Docs
+ New
304 components found
Filter by: All Components Labels Dependencies Size
PRIMENG / listbox v8.0.2
Listbox is used to select one or more values from a list of ...
MIT 7 KB
PRIMENG / blockui v8.0.2
BlockUI can either block other components or the whole page.
MIT 4 KB
PRIMENG / inplace v8.0.2
Inplace provides an easy to do editing and display at the sa...
MIT 5 KB
PRIMENG / fieldset v8.0.2
Fieldset is a grouping component with the optional content t...
MIT 3 KB
PRIMENG / rating v8.0.2
Rating components is a star based selection input.
MIT 1 KB
PRIMENG /
PRIMENG /
PRIMENG /
PRIMENG /
NGX-BOOTSTRAP /

Bit: easily share components across apps as a team

Angular lets you build your apps through modular components, which are the most basic UI building block of an Angular app. When building multiple Angular projects or applications, components can be shared and reused between them, to speed development and build better modular apps.

In this post, we'll see how to share your Angular components between different projects. We will use bit (GitHub) to seamlessly isolate and pack components from existing projects, and bit.dev to share them between multiple projects and applications. When done, your components will also be available for your team to discover, use and even develop anywhere.

You can learn more about sharing NG components using Bit here, or try the hands-on tutorial for yourself. Feel free to keep reading below to get started, and don't hesitate to ask anything :)

Why use Bit to share Angular components?

Bit is an open-source tool for sharing and collaborating on components.

Share reusable code components as a team · Bit

Easily share reusable components between projects and applications to build faster as a team. Collaborate to develop...

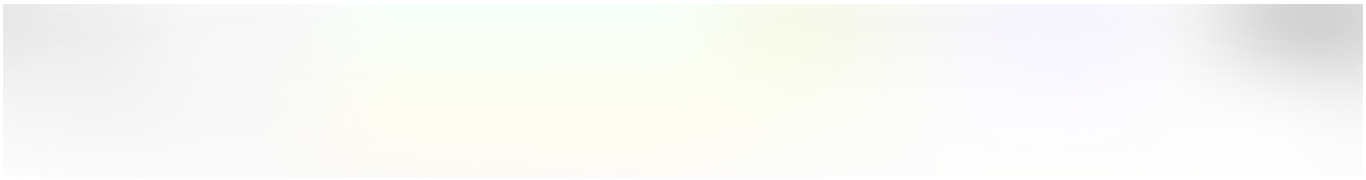
bit.dev

Using Bit you can quickly isolate components from any project, with all their setup and dependencies, and use them across different projects and apps. This means more components shared, and less overhead for the devs sharing them.

Read:

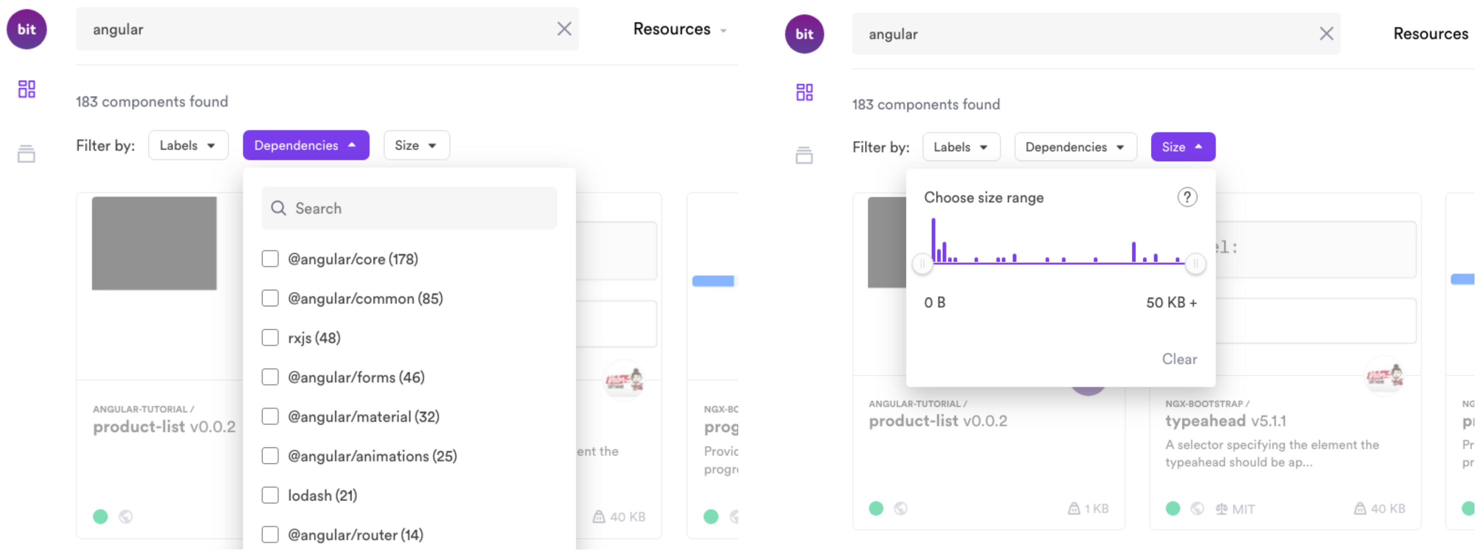
- **How we Build Our Design System**
- **How we Build Micro Frontends**

For example, here is the popular ngx-bootstrap library on GitHub. Using Bit, and in a few minutes, it's shared as a collection of components in bit.dev. You can do the same for your team's components, to collaborate and build apps.



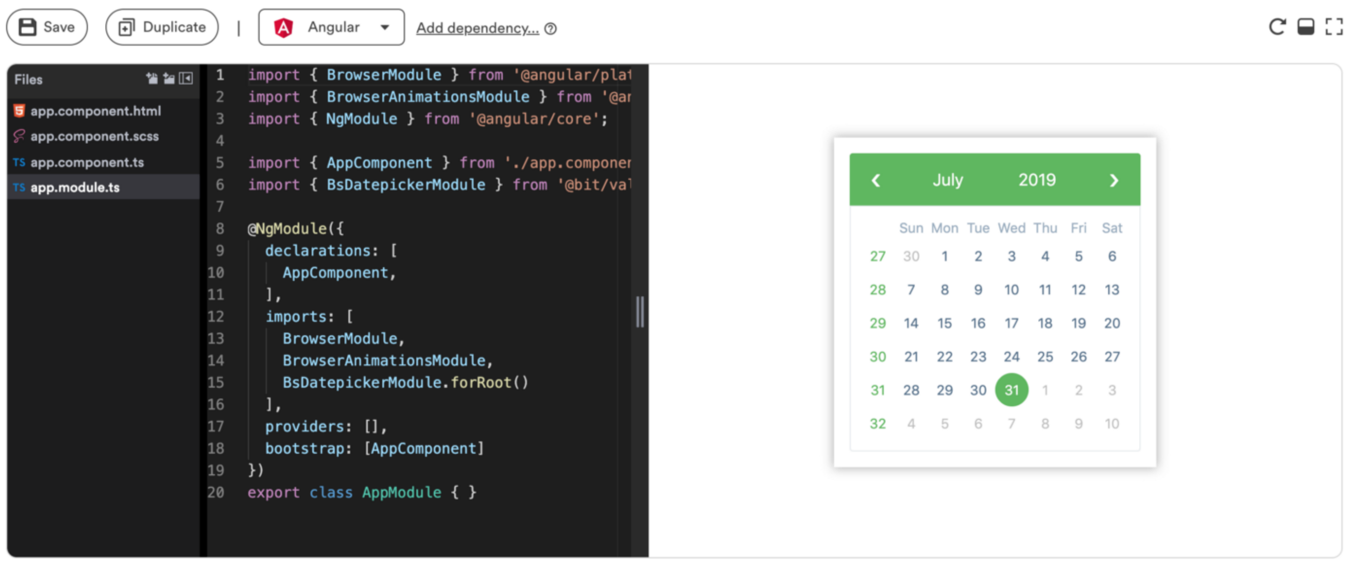
Angular components: easily share across projects

You can search and discover your components, view visual snapshots and filters components by bundle-size and dependencies to find the right code.



Search components by context, dependencies, and bundle-size

You can try each component in a live playground before choosing to use it in your project. This is useful for collaborating on many components as a team.



```
import { BrowserModule } from '@angular/plat
import { BrowserAnimationsModule } from '@ar
import { NgModule } from '@angular/core';

import { AppComponent } from './app.componer
import { BsDatepickerModule } from '@bit/val

@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    BrowserAnimationsModule,
    BsDatepickerModule.forRoot()
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Try each component hands-on before installing it

Then, you can install each component in your project with NPM/Yarn, and even use Bit to develop and update it right from any consuming project (:0).

Go ahead and get started, or explore components in the bit.dev community.

Let's get started

Let's see a hands-on example of sharing a component between two Angular applications. The same goes for 100 components, and it's just as easy.

Note that it's recommended to **wrap each component in a ngModule**. You can find more recommended guidelines and best practices here, and here's a useful tutorial to help you get started with a demo application:

Bit with Angular Tutorial

Angular support is in public beta. Everything should work fine, but please do share any issues and feedback…

docs.bit.dev

Setup Bit

To get started, go ahead and install bit-cli, then head over to the project from which to share the components, and initialize a bit workspace.

```
$ npm install bit-bin -g
$ cd project-directory
$ bit init
```

Then, head over to **bit.dev and create a free account**. Then, **create a collection** to host your shared components. For this tutorial, you can name your new shared collection "angular-tutorial".

Once done, authenticate bit-cli to your bit.dev account.

```
$ bit login
```

Share an Angular component

Ok, now you're all ready to get started. To help you get started, here's a hands-on tutorial for sharing an Angular component between two applications.

Clone the demo app

This tutorial is based on Angular 8. To run this tutorial, clone and setup the Angular tutorial project: https://github.com/teambit/bit-angular-tutorial

```
$ git clone https://github.com/teambit/bit-angular-tutorial
$ cd bit-angular-tutorial
$ npm install
```

We will share the **product-list component** from the Angular tutorial project, and use it in another application. When done, the component will be available in bit.dev for your team to discover, use and develop in their own projects.

Track the product-list component

Bit's commands are very much Git-like, for ease of use.

Let's use the `bit add` command to track the product-list component. We need to let Bit know where is the component and which files are related to it. As all the files are located under the product-list directory, the simplest way is to add all files in the directory to your component. Bit will name it accordingly.

For Angular components, we also need to specify the component entry point, which in most cases will be the file containing the ngModule. In this case, it is the `product-list.module.ts` file.

Run the following command:

```
$ bit add src/app/product-list  --main src/app/product-list/product-
list.module.ts

tracking component product-list:
added src/app/product-list/product-list.component.css
added src/app/product-list/product-list.component.html
added src/app/product-list/product-list.component.ts
added src/app/product-list/product-list.module.ts
added src/app/product-list/products.ts
```

That's it. Bit identified the component with all its files, defined any dependencies as part of the component and started tracking it.

You can make sure that all the files required for the component were properly added by using `bit status`:

```
$ bit status
new components
```

```
> product-list ... ok
```

> Note: In this case the products.ts file was added to the component. If it is used by other components too, turn it into a component which will become a dependency of the product-list and other components.

Install the Angular compiler

So far, we have provided Bit with the source file of the component. But in order to consume the files in other projects, the component needs to be built.

Bit has a large collection of compilers to choose from (you can add your own). For building the Angular component, you'll need the Angular compiler.

compilers/angular - envs · Bit

Please note, this environment is still experimnental, use this with caution. Works with bit version >= 14.2.3 Bit…

bit.dev

Install the compiler by running this command inside the tutorial repository:

```
$ bit import bit.envs/compilers/angular --compiler

the following component environments were installed
- bit.envs/compilers/angular@0.1.2
```

> Versions may slightly vary when you run the tutorial.

The Angular compiler is now set as the default compiler for the Bit workspace. You can check the package.json and verify that the compiler is installed by locating the following entry in the Bit section:

```
"env": {
      "compiler": "bit.envs/compilers/angular@0.1.2"
    },
```

To build your component, run this command inside your project:

```
$ bit build
```

This results in the component name (product-list) followed by a list of file names. Those are the built files of your the component 👍

Export the component to bit.dev

With the component now properly built, it is now time to share it with the world. Components are versioned according to semver standards. To Tag your component with a version, run the following command:

```
$ bit tag --all 0.0.1
1 component(s) tagged

new components
     > product-list@0.0.1
```

This command tags all the components that are currently staged in Bit’s workspace. In our case, it’s only the product-list component.

You can run `bit status` to validate the versioned components.

```
$ bit status
staged components

     > product-list. versions: 0.0.1 ... ok
```

The important thing to notice here is that the component is considered `staged`. That means that it is now ready to be exported. Next, you can `export` the component to your bit.dev collection:

```
$ bit export <username>.angular-tutorial
exported 1 components to collection <username>.angular-tutorial
```

The component is now visible in your collection on bit.dev. You can access it in `https://bit.dev/<username>/angular-tutorial`. You can also visit the component created for this demo on: https://bit.dev/bit/angular-tutorial

Checking now the component's status will no longer display the component in the local workspace, as it is now hosted in the remote bit.dev collection:

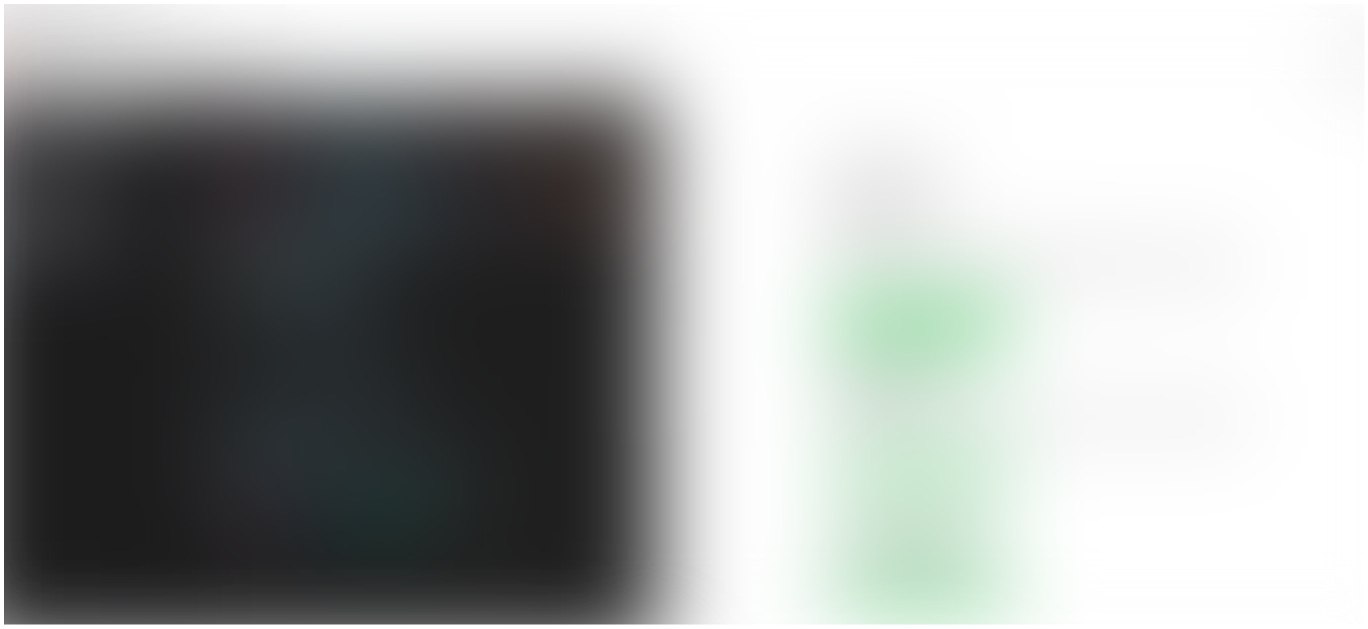
```
$ bit status
nothing to tag or export
```

If you want to see all your components and their versions you can run:

```
$ bit list
```

Right now, the component code is still in your local project (and should be committed to your source control), but it is also available for other projects.

Preview the component in bit.dev



The Angular component is also available on the bit.dev cloud. Go to `https://bit.dev` and log into your account (if you are not logged in yet):

1. Select the collections navigator on the left panel and select collections.
2. Click on your collection you'll see your component product-list.
3. Click on the product-list component to see it's live playground.

The component playground provides you with a basic Angular app (you may recognize the sample as the app provided by Angular CLI). We will modify this application to display the product-list component.

Edit the component example

The component is automatically recognized as an Angular component. We will modify the `app.module.ts` file to import our product-list module.

Change this line:

```
import ProductList from '@bit/.angular-tutorial.product-list';
```

Into this line:

```
import {ProductListModule} from '@bit/.angular-tutorial.product-
list';
```

Add the module to the Module's `imports` section:

```
imports: [
        BrowserModule,
        ProductListModule
    ],
```

The `app.module.ts` should look like this:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { ProductListModule } from '@bit/bit.angular-tutorial.product-
list';

@NgModule({
    declarations: [
        AppComponent,
    ],
    imports: [
        BrowserModule,
        ProductListModule
    ],
    providers: [],
    bootstrap: [AppComponent]
})
export class AppModule { }
```

Modify the `app.component.html` and replace its contents with:

```
<app-product-list></app-product-list>
```

Save the example. In a few seconds, you will see the component rendered in the playground. You can view an example here.

Install the component in a different project

You are now going to create another Angular application and use the product-list component in this new application.

Create a New Angular Application

The fastest way to do that is by using the Angular CLI to generate a new Application. Switch to a new directory.

If you already have the Angular-cli installed globally you can just run:

```
$ ng new my-new-app
```

If you do not have the angular-cli installed globally and you do not want to install it, you can use `npx` to install it temporarily:

```
$ npx --package @angular/cli ng new my-new-app
```

For the purpose of this demo, you can skip router and select css for styling. In your terminal, switch to the `my-new-app` directory.

Install the component using npm/yarn

Use your favorite package installer (npm or yarn) to install the component. The component is stored in the bit.dev registry, so the full path to the component will be: `@bit/<username>.<collection name>.<component name>`

Run the install command using npm:

```
$ npm install @bit/<username>.angular-tutorial.product-list --save
```

The component is now added to your `package.json`:

```
"@bit/<username>.angular-tutorial.product-list": "0.0.1"
```

Use the component in the new application

Now you can use the component in your code, as any import. Add it as a module to the top-level app module and use it on the app page. We will make the same changes in the code as we did on the playground in the application:

```
// app.module.ts
import { ProductListModule } from '@bit/<username>.angular-
tutorial.product-list';
```

Add the `ProductListModule` in your app module imports section:

```
imports: [
        BrowserModule,
        ProductListModule
    ],
```

All is left now is to add the product-list component to the app html file. You can replace all existing content with the line above, or add it on the html page.

```
// src/app/app.component.html
<app-product-list></app-product-list>
```

Last, but not least, run your application using Angular CLI:

```
$ npm run start
```

Awesome! you can now see the components list inside the newly created application. You've just shared a component from one project to another

> *Do the same for many components, and invite your team to collaborate…*

Modify the component from the consuming project

Bit has a unique ability: it lets you develop and update components from any project using them. This is very useful to speed development, as it enables you and your team members to collaborate and suggest updates as needed.

We are going to make a change to the component and export it back to the collection. We will add a **View** button to the product list. For simplicity, it will only show an alert saying the product is viewed.

Import the component

So far, the product-list component was only installed (in its built form) in our project. Now, we want to import the code into our project and make changes.

In order to import the component, initiate a Bit workspace in the new `my-new-app` project:

```
$ bit init
```

Then run the following command (also available on the component page):

```
$ bit import <username>.angular-tutorial/product-list

bit import bit.angular-tutorial/product-list
successfully imported one component
- added bit.angular-tutorial/product-list new versions: 0.0.1,
currently used version 0.0.1
```

> Notifications on missing core dependencies are ok. You should already have those packages in your project.

You will get a message that the `@angular/core` and `@angular/common` are peer dependencies. This is ok, as your `my-new-app` project already contains them.

Here is what happened:

- A new top-level components folder is created that includes the code of the component and its compiled code and node*modules (in this case the node*modules are empty, as all of your node_modules are peer dependencies and are taken from the root project.
- The `.bitmap` file was modified to include the reference to the component
- The package.json file is modified to point to the files rather than the remote package. Your `package.json` now displays:

```
"@bit/.angular-tutorial.product-list": "file:./components/product-
list"
```

Start your application to make sure it still works (that is true, no changes are required. Bit takes care of everything).

Update the code

Let's modify the product-list component. Change the `components/product-list/product-list.component.ts` to include the following method:

```
view() {
    window.alert('The product has been viewed!');
 }
```

Change the `components/product-list/product-list.component.html` to include the following part after the `share button`:

```
<button (click)="view()">
        View
    </button>
```

Change the css to contain a margin on the `button`:

```
margin: 4px;
```

Run the Angular application:

```
$ npm run start
```

The app is not yet changed. The Bit components are compiled by the bit compiler. In a separate terminal, run the `bit build` command to compile the changes. You should see that the compiler is installed:

```
successfully installed the bit.envs/compilers/angular@0.1.2 compiler
```

Followed by a successful compilation of all the files.

Your angular project will refresh, and you can now see the changed component with the `view button`.

> In a real project, it is recommended to commit those changes to your GitHub repository.

Export the changes

Next, export the changes done to the component back to bit.dev.

```
$ bit status
```

The product-list component was modified:

```
modified components

     > product-list ... ok
```

Tag and export the component as a new version. By default this is a semver `patch` version:

```
$ bit tag product-list
1 component(s) tagged

changed components
(components that got a version bump)
     > <username>.angular-tutorial/product-list@0.0.2
```

Export it back to the collection:

```
$ bit export <username>.angular-tutorial
exported 1 components to scope <username>.angular-tutorial
```

Head to the component page on bit.dev, here you can see that the component has a new version. The changes are also visible on the component playground.

Update the changes in the original project

In this last stage, you are going to import the changes to the original project, switch back to `Angular-tutorial`.

Import changes

Run `bit import` to see if any components were changed (similar to doing git pull to check git changes).

we will see that the product-list component was changed and a new version exists:

```
$ bit import
successfully imported one component
- updated <username>.angular-tutorial/product-list new versions:
0.0.2
```

The component is downloaded but is not yet changed. Check the workspace status, you will get the following:

```
$ bit status
pending updates

    > <username>.angular-tutorial/product-list current: 0.0.1 latest:
0.0.2
```

Checkout

Merge the changes done to the component to your project. The structure of the command is `bit checkout <version> <component>`. So you run:

```
$ bit checkout 0.0.2 product-list
successfully switched <username>.angular-tutorial/product-list to
version 0.0.2
updated src/app/product-list/product-list.component.css
updated src/app/product-list/product-list.component.html
updated src/app/product-list/product-list.component.ts
updated src/app/product-list/product-list.module.ts
updated src/app/product-list/products.ts
```

Bit is performing a git merge, so the code from the updated component will now be merged into your code. Run the app again, to see it is working properly with the updated component:

```
$ npm run start
```

That is it. A change was moved and synced between the two projects. Your application is now running with an updated component….

Some guidelines for Angular with bit…

Bit's team worked together with the Angular team to define some best practices for sharing reusable components. You can find these best practices in this link here. If you have any questions don't hesitate, just ask for help :)

Conclusion

Sharing component between your Angular applications means you can build modular applications that leverage code-reuse and composition, therefore they are easier and faster to build while also simpler to maintain.

Bit lets your team avoid the overhead around publishing components at any scale, provides a universal discovery hub for your team, and lets you modify and update components from any of your projects to collaborate together.

The reusable nature of angular components plays well with Bit's advantages, to help you and your team successfully share code. It works well with a component library or directly between your apps, so it's up to you…

Happy sharing! 🍺

Learn more

How We Build a Design System

Building a design system with components to standardize and scale our UI development process.

blog.bitsrc.io

How We Build Micro Frontends

Building micro-frontends to speed up and scale our web development process.

blog.bitsrc.io

Announcing Bit with Angular Public Beta

Special thanks to the awesome Angular team for working together on making this happen 🤝

blog.bitsrc.io

The State of Angular in 2019

A detailed overview of the current state of Angular, from the latest features to the hottest topics and trends that you…

blog.bitsrc.io

11 Angular Component Libraries You Should Know In 2019

11 popular Angular component libraries for building your Angular app in 2019.

blog.bitsrc.io

JavaScript Angular Bit Web Development Shared Components