// src/app/components/my-button/my-button.stories.ts

import { moduleMetadata, Meta, StoryObj } from '@storybook/angular';

import { MyButtonComponent } from './my-button.component';

const meta: Meta<MyButtonComponent> = {

title: 'Components/My Button',

component: MyButtonComponent,

decorators: [

moduleMetadata({

declarations: [],

imports: [], // add any modules if needed

}),

],

tags: ['autodocs'], // optional, for docs

};

export default meta;

type Story = StoryObj<MyButtonComponent>;

export const Primary: Story = {

args: {

label: 'Click Me',

},

};

Absolutely! Let's walk through that code **line by line** so you understand exactly how it works. You're defining a **Storybook story** for an Angular component (MyButtonComponent). Here's the breakdown:

**📄 File: my-button.stories.ts**

import { moduleMetadata, Meta, StoryObj } from '@storybook/angular';

import { MyButtonComponent } from './my-button.component';

**👉 What's happening here?**

* moduleMetadata: Allows you to define extra Angular metadata like imports, providers, or declarations — just like in @NgModule. It's useful when your component depends on other modules (like Material components, FormsModule, etc.).
* Meta: A generic type that describes the metadata for the component's story (title, component, etc.).
* StoryObj: Used to define individual "stories" — i.e., different configurations of your component.

const meta: Meta<MyButtonComponent> = {

title: 'Components/My Button',

component: MyButtonComponent,

decorators: [

moduleMetadata({

declarations: [],

imports: [], // add any modules if needed

}),

],

tags: ['autodocs'],

};

**👉 What's happening here?**

You're setting up the **main metadata for Storybook** to understand how to render your component.

* title: This defines how it appears in the Storybook UI — like a folder path. So this will show up as **"Components / My Button"** in the sidebar.
* component: The actual Angular component you're writing stories for.
* decorators: Optional array of enhancements. Here, you're using moduleMetadata to simulate an Angular module. If MyButtonComponent depends on other modules (e.g., CommonModule, FormsModule), you'd add them in the imports array.
* tags: ['autodocs']: Optional. Used by Storybook Docs to auto-generate documentation pages.

export default meta;

**👉 This exports the metadata**

Storybook needs this default export to register your component’s story and metadata.

type Story = StoryObj<MyButtonComponent>;

**👉 Typing shortcut**

You're defining a **type alias** called Story to make it easier to reuse the StoryObj<MyButtonComponent> type when writing multiple stories.

export const Primary: Story = {

args: {

label: 'Click Me',

},

};

**👉 This is your first story**

* Primary: The name of the story — this is like a variation or example of how your component can look.
* args: These are the inputs (@Input()s) passed to your component.
  + For example, if your MyButtonComponent has an input like @Input() label: string, this sets it to "Click Me".

This will render your button component in Storybook with the label "Click Me".

**🔁 If You Had More Variants:**

export const Secondary: Story = {

args: {

label: 'Cancel',

},

};

Each story shows a different state/config of your component.

**✅ Summary**

| **Part** | **Purpose** |
| --- | --- |
| Meta | Describes the component, its title, and needed modules |
| StoryObj | Describes a specific version or variant of the component |
| args | Values passed into component inputs (@Input()) |
| moduleMetadata | Simulates an Angular module to inject needed declarations/imports |
| title | Storybook UI label ("folder" in sidebar) |

Let me know if you want to test this live, or add things like click events (@Output()), style variations, or Material components!

Awesome! Let’s do a **full walk-through with a working component**, and I’ll show you how to include @Input() and @Output() as well.

**🧱 Step-by-Step: Make MyButtonComponent Story-Ready**

**✅ 1. Create the Component**

ng generate component components/my-button

**✅ 2. Edit my-button.component.ts**

Update it to include inputs and an output for click:

import { Component, Input, Output, EventEmitter } from '@angular/core';

@Component({

selector: 'app-my-button',

template: `<button (click)="onClick()">{{ label }}</button>`,

styles: [`

button {

padding: 8px 16px;

border: none;

background-color: #1976d2;

color: white;

font-size: 16px;

border-radius: 4px;

cursor: pointer;

}

`]

})

export class MyButtonComponent {

@Input() label = 'Default';

@Output() clicked = new EventEmitter<void>();

onClick() {

this.clicked.emit();

}

}

**✅ 3. Create the Story File**

File: src/app/components/my-button/my-button.stories.ts

import { Meta, StoryObj, moduleMetadata } from '@storybook/angular';

import { MyButtonComponent } from './my-button.component';

import { CommonModule } from '@angular/common';

const meta: Meta<MyButtonComponent> = {

title: 'Components/My Button',

component: MyButtonComponent,

decorators: [

moduleMetadata({

imports: [CommonModule], // Add any necessary Angular modules here

}),

],

tags: ['autodocs'],

};

export default meta;

type Story = StoryObj<MyButtonComponent>;

export const Primary: Story = {

args: {

label: 'Click Me',

},

};

export const WithClickEvent: Story = {

args: {

label: 'Click Me',

},

// Adding an action logger

argTypes: {

clicked: { action: 'button clicked!' },

},

};

**✅ 4. Start Storybook**

npm run storybook

You should now see:

* Primary: A basic button labeled "Click Me"
* WithClickEvent: A button that logs "button clicked!" in the **Actions** panel when clicked

**🌟 Bonus: Styling & Customization**

You can pass more inputs like color, disabled, size, etc. and update the story to test those variations.

Let me know if you want to:

* Add Tailwind or Material styling
* Make it a loading button with spinner
* Pass icons or templates inside the button
* Use Storybook controls to dynamically change props

Just say the word 👇