



# StoryBook with Angular

[bit.ly/w-sb-angular](https://bit.ly/w-sb-angular)

**bit.ly/w-sb-angular**

## ERROR STATE

What happens when one prop is missing?

If network fails?

Long Names

DARK MODE

BREAKPOINT  
BREAKPOINT  
BREAKPOINT

Internationalization

PHABLETS?

## IS THE SPACING CORRECT?

POOR FORMATTING

## ACCESSIBILITY

## LOADING STATE

## LOADING PERFORMANCE

## BAD DATA

Mobile Safari?

## COOL, BUT WHAT ABOUT...

Develop

Document

Test

# **Exercises**

**Story for a presentational component**

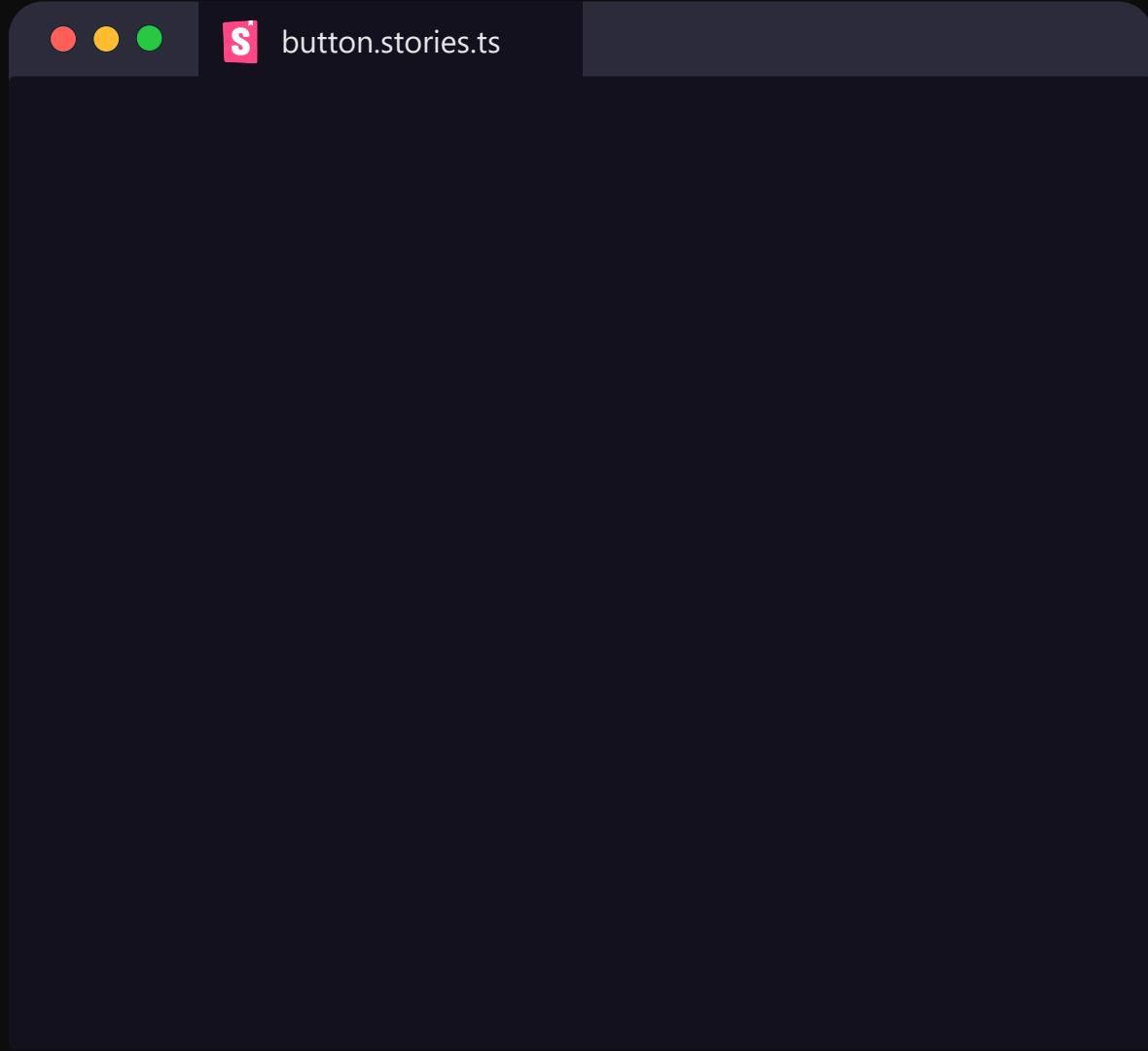
**Story for a container component**

**Creating a guideline**

**Adding custom documentation for a component**

**Extra – Tests and Accessibility addons**

**Button!**





button.stories.ts

```
import { Meta, StoryObj, argsToTemplate } from '@storybook/angular';
import { ButtonComponent } from './button.component';

const meta: Meta<ButtonComponent> = {
  title: 'Shared/Components/Button',
};


```



button.stories.ts

```
import { Meta, StoryObj, argsToTemplate } from '@storybook/angular';
import { ButtonComponent } from './button.component';

const meta: Meta<ButtonComponent> = {
  title: 'Shared/Components/Button',
  component: ButtonComponent,
};


```



button.stories.ts

```
import { Meta, StoryObj, argsToTemplate } from '@storybook/angular';
import { ButtonComponent } from './button.component';

const meta: Meta<ButtonComponent> = {
  title: 'Shared/Components/Button',
  component: ButtonComponent,
  render: (args) => ({
    props: args,
    template: `<ui-button ${argsToTemplate(args)}>Button!</ui-button>`,
  }),
};
```



**S** button.stories.ts

```
import { Meta, StoryObj, argsToTemplate } from '@storybook/angular';
import { ButtonComponent } from './button.component';

const meta: Meta<ButtonComponent> = {
  title: 'Shared/Components/Button',
  component: ButtonComponent,
  render: (args) => ({
    props: args,
    template: `<ui-button ${argsToTemplate(args)meta;
type Story = StoryObj<ButtonComponent>;
```



## S button.stories.ts

```
import { Meta, StoryObj, argsToTemplate } from '@storybook/angular';
import { ButtonComponent } from './button.component';

const meta: Meta<ButtonComponent> = {
  title: 'Shared/Components/Button',
  component: ButtonComponent,
  render: (args) => ({
    props: args,
    template: `<ui-button ${argsToTemplate(args)}>Button!</ui-button>`,
  }),
};

export default meta;
type Story = StoryObj<ButtonComponent>;

export const Default: Story = {};
```

button.stories.ts

```
import { Meta, StoryObj, argsToTemplate } from '@storybook/angular';
import { ButtonComponent } from './button.component';

const meta: Meta<ButtonComponent> = {
  title: 'Shared/Components/Button',
  component: ButtonComponent,
  render: (args) => ({
    props: args,
    template: `<ui-button ${argsToTemplate(args)}>Button!</ui-button>`,
  }),
};

export default meta;
type Story = StoryObj<ButtonComponent>;

export const Default: Story = {};
```

Storybook

Find components /

SHARED

Components

Button

Documentation

Default

Primary

Secondary

Primary And Large

Controls 6 Actions Visual Tests

Name Control

PROPERTIES

buttonClass Set string

size Set string

type Set string

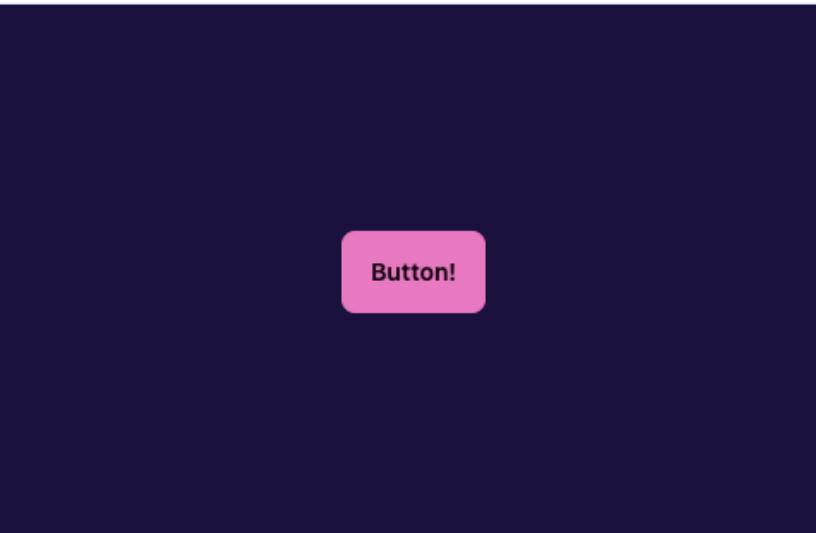
OUTPUTS

click Set object

INPUTS

disabled Set boolean

METHODS



The screenshot shows the Storybook interface for a 'Button' component. On the left, the code for 'button.stories.ts' is displayed, defining a story named 'Default' for the 'ButtonComponent'. The Storybook sidebar on the right lists the 'Components' section, which contains the 'Button' component. Under 'Button', there are four stories: 'Documentation', 'Default' (which is selected and highlighted in blue), 'Primary', 'Secondary', and 'Primary And Large'. Below the stories, the 'Properties' section is visible, showing controls for 'buttonClass', 'size', and 'type', each with a 'Set string' button. The 'Outputs' section shows a 'click' output with a 'Set object' button. The 'Inputs' section shows a 'disabled' input with a 'Set boolean' button. At the bottom, there are tabs for 'Controls' (with 6 items), 'Actions', and 'Visual Tests'. A preview window on the right shows a single pink button with the text 'Button!'.

**S** button.stories.ts

```
import { Meta, StoryObj, argsToTemplate } from '@storybook/angular';
import { ButtonComponent } from './button.component';

const meta: Meta<ButtonComponent> = {
  title: 'Shared/Components/Button',
  component: ButtonComponent,
  render: (args) => ({
    props: args,
    template: `<ui-button ${argsToTemplate(args)}>Button!</ui-button>`,
  }),
};

export default meta;
type Story = StoryObj<ButtonComponent>;

export const Default: Story = {};

export const Primary = {
  args: {
    size: 'md',
    type: 'primary',
  },
};
```

**S Storybook**

Find components /

SHARED Components

Button Documentation Default Primary Secondary Primary And Large

Controls 6 Actions Visual Tests

Name Control

PROPERTIES

size md

type primary

buttonClass Set string

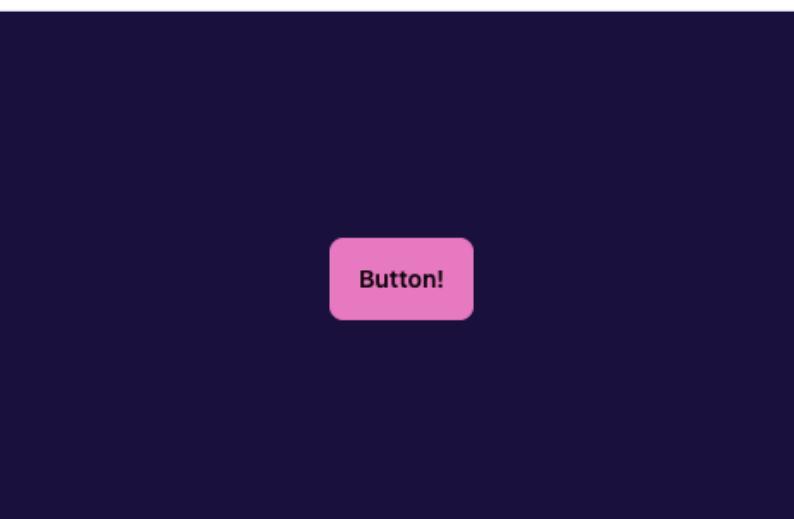
OUTPUTS

click Set object

INPUTS

disabled Set boolean

METHODS



**S** button.stories.ts

```
import { Meta, StoryObj, argsToTemplate } from '@storybook/angular';
import { ButtonComponent } from './button.component';

const meta: Meta<ButtonComponent> = {
  title: 'Shared/Components/Button',
  component: ButtonComponent,
  render: (args) => ({
    props: args,
    template: `<ui-button ${argsToTemplate(args)}>Button!</ui-button>`,
  }),
};

export default meta;
type Story = StoryObj<ButtonComponent>;

export const Default: Story = {};

export const Primary = {
  args: {
    size: 'md',
    type: 'primary',
  },
};

export const Secondary = {
  args: {
    type: 'secondary',
  },
};
```

**S Storybook**

Find components /

SHARED

Components

Button

- Documentation
- Default
- Primary
- Secondary**
- Primary And Large

Controls 6 Actions Visual Tests

Name	Control
type	secondary
buttonClass	Set string
size	Set string

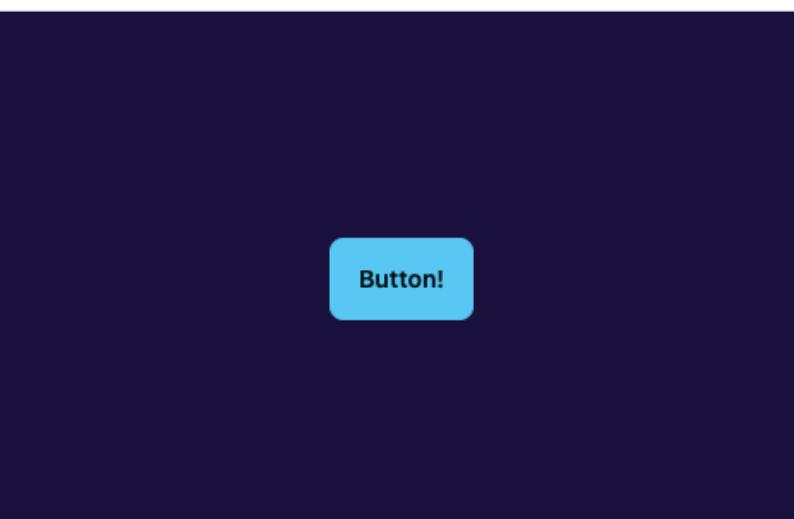
OUTPUTS

click	Set object
-------	------------

INPUTS

disabled	Set boolean
----------	-------------

METHODS



```
import { Meta, StoryObj, argsToTemplate } from '@storybook/angular';
import { ButtonComponent } from './button.component';

const meta: Meta<ButtonComponent> = {
  title: 'Shared/Components/Button',
  component: ButtonComponent,
  render: (args) => ({
    props: args,
    template: `<ui-button ${argsToTemplate(args)}>Button!</ui-button>`,
  }),
};

export default meta;
type Story = StoryObj<ButtonComponent>;

export const Default: Story = {};

export const Primary = {
  args: {
    size: 'md',
    type: 'primary',
  },
};

export const Secondary = {
  args: {
    type: 'secondary',
  },
};

export const PrimaryAndLarge = {
  args: {
    size: 'lg',
    type: 'primary',
  },
};
```

The screenshot shows the Storybook interface with the following details:

- Header:** Storybook logo, search bar with placeholder "Find components /", and various navigation icons.
- Sidebar:** A tree view of components under "Components".
  - "Button" component has four stories:
    - Documentation
    - Default
    - Primary
    - Secondary
  - "Primary And Large" story is currently selected, highlighted with a blue background.
- Preview Area:** A pink button with the text "Button!" inside it.
- Controls Panel:** Shows the current state of controls for the selected story.

Name	Control
size	lg
type	primary
buttonClass	Set string
- Properties Panel:** A table showing properties and their values.

PROPERTIES	
Name	Value
size	lg
type	primary
buttonClass	Set string
- Outputs Panel:** A table showing outputs and their values.

OUTPUTS	
Name	Type
click	Set object
- Inputs Panel:** A table showing inputs and their values.

INPUTS	
Name	Type
disabled	Set boolean
- Methods Panel:** A table showing methods and their values.

METHODS	
---------	--

## Exercise

# Story for a presentational component

- Add Story for Pagination component with the following options

```
{  
  totalItems: 100,  
  currentPage: 1,  
  itemsPerPage: 10,  
}
```

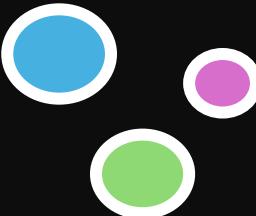
- Set the location of the story to be on 'Shared/Components'
- Add a story of 'WithFewItems' that has only 10 total items



# Develop



# Develop Atoms



storybook

Find components /

SHARED

Components

Button

Documentation

Default

Primary

Secondary

Primary And Large

Controls 6 Actions Visual Tests

Name Control

Properties

type secondary

buttonClass Set string

size Set string

Outputs

click Set object

Inputs

disabled Set boolean

METHODS

Button!

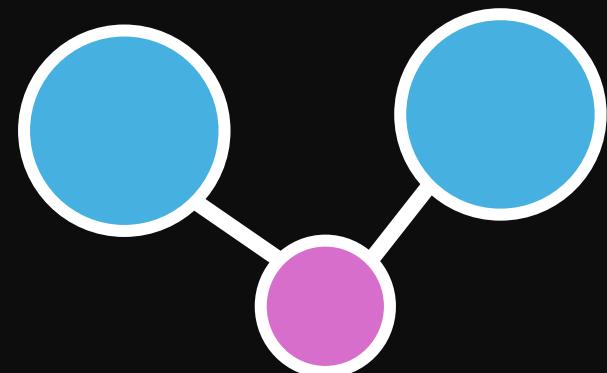
This screenshot shows a Storybook interface for developing atomic design systems. On the left, a large white box contains the text "Develop Atoms". Below it are three circular icons: one blue, one pink, and one green, arranged in a cluster. The main area is a dark blue Storybook preview window. Inside the preview, there is a single blue button with the text "Button!". To the right of the preview are several tabs: "Controls" (selected), "Actions", and "Visual Tests". Below these tabs is a table with columns for "Name" and "Control". Under the "Properties" section, there are fields for "type" (set to "secondary"), "buttonClass" (with a "Set string" button), and "size" (with a "Set string" button). Under the "Outputs" section, there is a field for "click" (with a "Set object" button). Under the "Inputs" section, there is a field for "disabled" (with a "Set boolean" button). At the bottom, there is a section labeled "METHODS". The sidebar on the left lists shared components under the "Components" section, with "Button" expanded. Within "Button", the "Secondary" variant is selected, highlighted with a blue background. Other variants listed include "Documentation", "Default", "Primary", and "Primary And Large".

# Develop



# Develop

# Molecules



storybook

Find components /

welcome

SHARED

Components

Button

- Documentation
- Default
- Primary
- Secondary
- Primary And Large

Pagination

- Documentation
- Default
- With Few Items

Previous Next

Controls 10 Actions Visual Tests

Name Control

INPUTS

totalItems 100

PROPERTIES

currentPage Edit string...

# Try States

# Try States

## Button

- Primary
- Secondary
- Primary and Large

## Pagination

- With 100 items
- With Few Items

The screenshot shows the Storybook interface with the following details:

- Toolbar:** Includes icons for refresh, search, and navigation.
- Header:** Storybook logo and settings icon.
- Search Bar:** "Find components" input field.
- Sidebar:** "SHARED" section expanded, showing "Components" folder. Inside "Components" are "Button" and several other items: Documentation, Default, Primary, Secondary (highlighted in blue), and Primary And Large.
- Preview Area:** A dark blue background with a single blue button labeled "Button!".
- Controls Panel:** Shows 6 controls. The first control is selected, showing properties for "type": "secondary", "buttonClass": "Set string", and "size": "Set string".
- Outputs:** "click" output set to "Set object".
- Inputs:** "disabled" input set to "Set boolean".
- Methods:** No methods are listed.

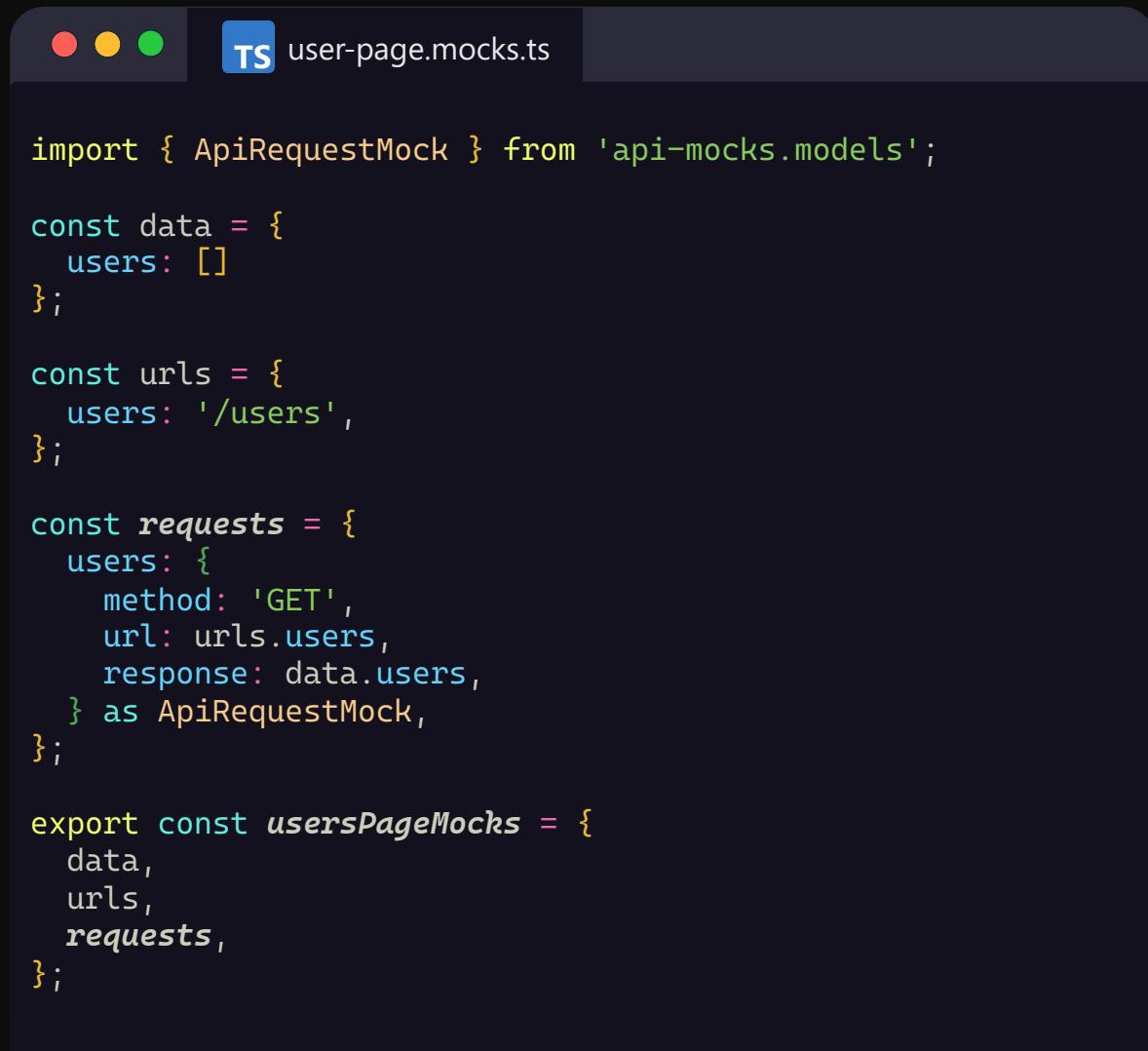
# Smart Component

# **Smart Component**

**Mocks File**

**API Service Mock**

# Mocks File



The image shows a screenshot of a code editor window titled "user-page.mocks.ts". The window has a dark theme with a header bar featuring three colored dots (red, yellow, green) and a "TS" icon. The main area contains the following TypeScript code:

```
import { ApiRequestMock } from 'api-mocks.models';

const data = {
  users: []
};

const urls = {
  users: '/users',
};

const requests = {
  users: {
    method: 'GET',
    url: urls.users,
    response: data.users,
  } as ApiRequestMock,
};

export const usersPageMocks = {
  data,
  urls,
  requests,
};
```

# Mocks File

● ● ● TS user-page.mocks.ts

```
import { ApiRequestMock } from 'api-mocks.models';

const data = {
  users: []
};

const urls = {
  users: '/users',
};

const requests = {
  users: {
    method: 'GET',
    url: urls.users,
    response: data.users,
  } as ApiRequestMock,
};

export const usersPageMocks = {
  data,
  urls,
  requests,
};
```

Data

# Mocks File

● ● ● TS user-page.mocks.ts

```
import { ApiRequestMock } from 'api-mocks.models';

const data = {
  users: []
};

const urls = {
  users: '/users',
};

const requests = {
  users: {
    method: 'GET',
    url: urls.users,
    response: data.users,
  } as ApiRequestMock,
};

export const usersPageMocks = {
  data,
  urls,
  requests,
};
```

Data

URLs

# Mocks File

● ● ● TS user-page.mocks.ts

```
import { ApiRequestMock } from 'api-mocks.models';

const data = {
  users: []
};

const urls = {
  users: '/users',
};

const requests = {
  users: {
    method: 'GET',
    url: urls.users,
    response: data.users,
  } as ApiRequestMock,
};

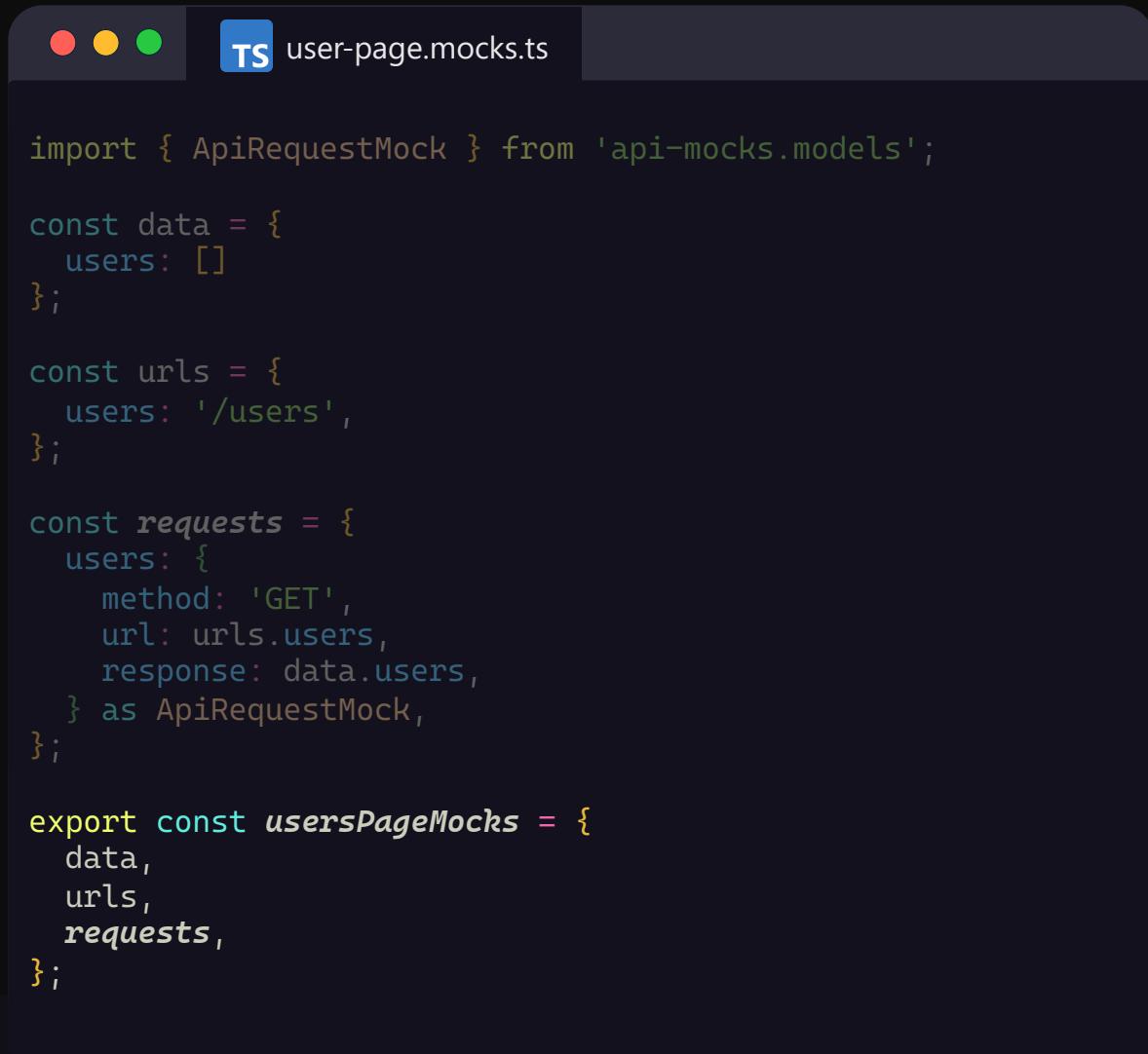
export const usersPageMocks = {
  data,
  urls,
  requests,
};
```

Data

URLs

Requests

# Mocks File



The screenshot shows a code editor window with a dark theme. The title bar says "user-page.mocks.ts". The code in the editor is:

```
import { ApiRequestMock } from 'api-mocks.models';

const data = {
  users: []
};

const urls = {
  users: '/users',
};

const requests = {
  users: {
    method: 'GET',
    url: urls.users,
    response: data.users,
  } as ApiRequestMock,
};

export const usersPageMocks = {
  data,
  urls,
  requests,
};
```

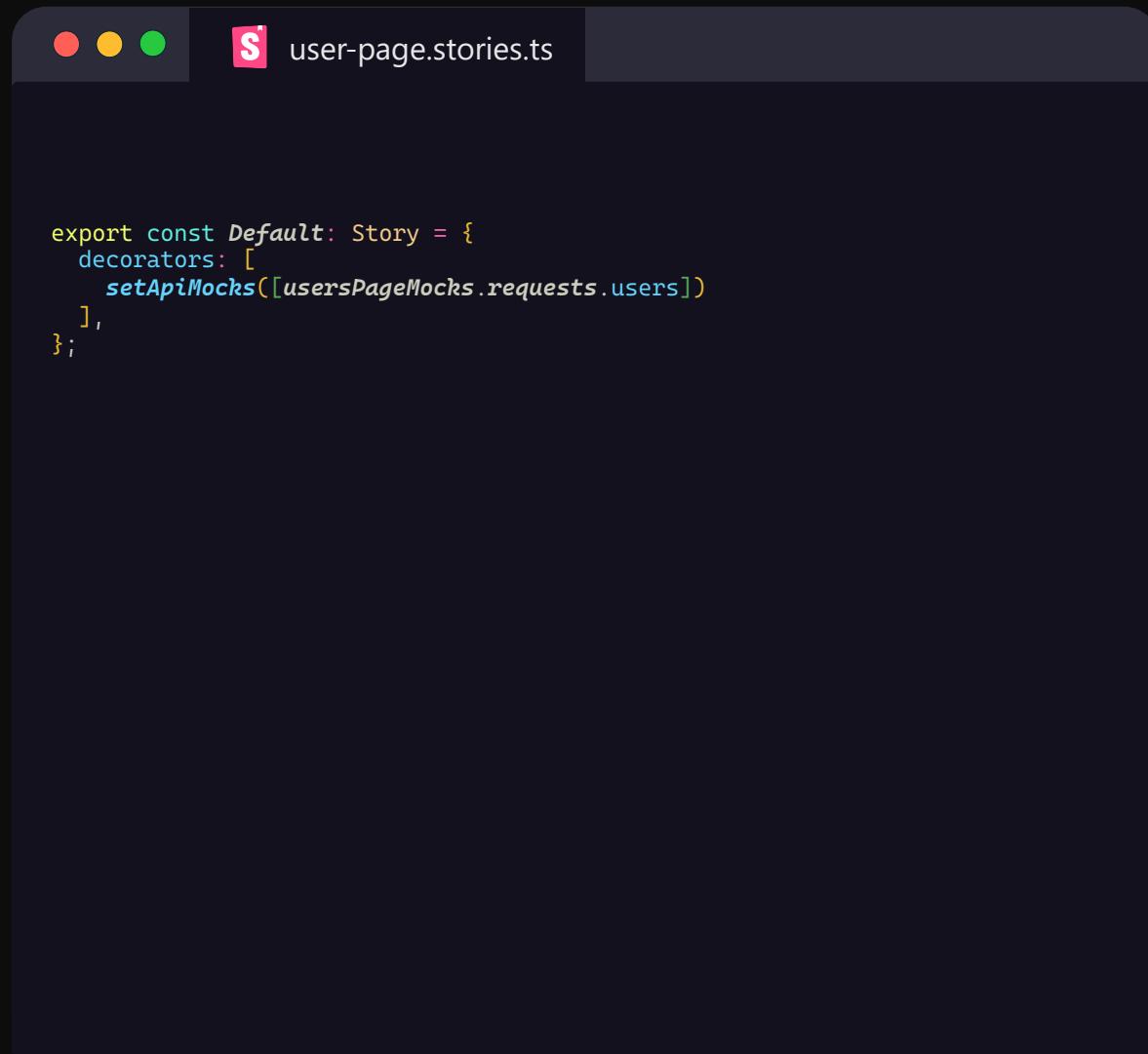
Data

URLs

Requests

User Page Mocks

# API Service Mock



A screenshot of a code editor window titled "user-page.stories.ts". The window has a dark theme with light-colored text. The code inside the file is:

```
export const Default: Story = {
  decorators: [
    setApiMocks([usersPageMocks.requests.users])
  ],
};
```

# **Smart Component**

**Mocks File**

**API Service Mock**

## **Exercise**

# **Story for a smart component**

- Add Story for Users Page
- Set the location of the story to be on 'Users/Users Page'
- Add a story of that uses the user page mocks with 'setApiMocks'

5:00

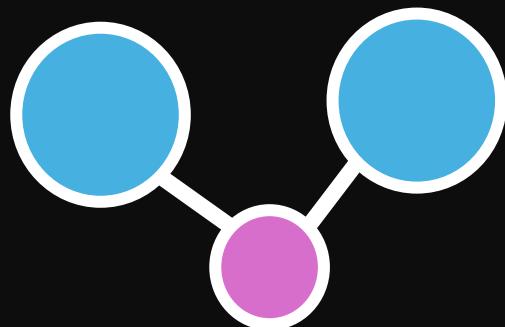
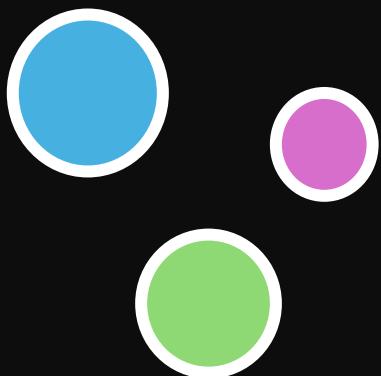


2-story-for-smart-component

# Develop



# Develop



# Smart Component

# **Smart Component**

**Mocks File**

**API Service Mock**

# Smart Component

## Mocks File

API Mock





# Mock Service Worker

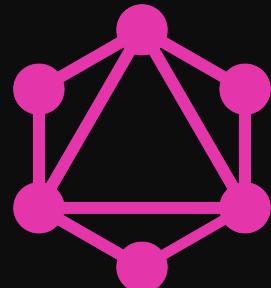
**API mocking library that allows you to write client-agnostic mocks and reuse them across any frameworks, tools, and environments**





# Mock Service Worker

**API mocking library that allows you to write client-agnostic mocks and reuse them across any frameworks, tools, and environments**



# Smart Component

**Why?**

**Keep State**  
**Request Data**  
**Business Rules**

**Names**

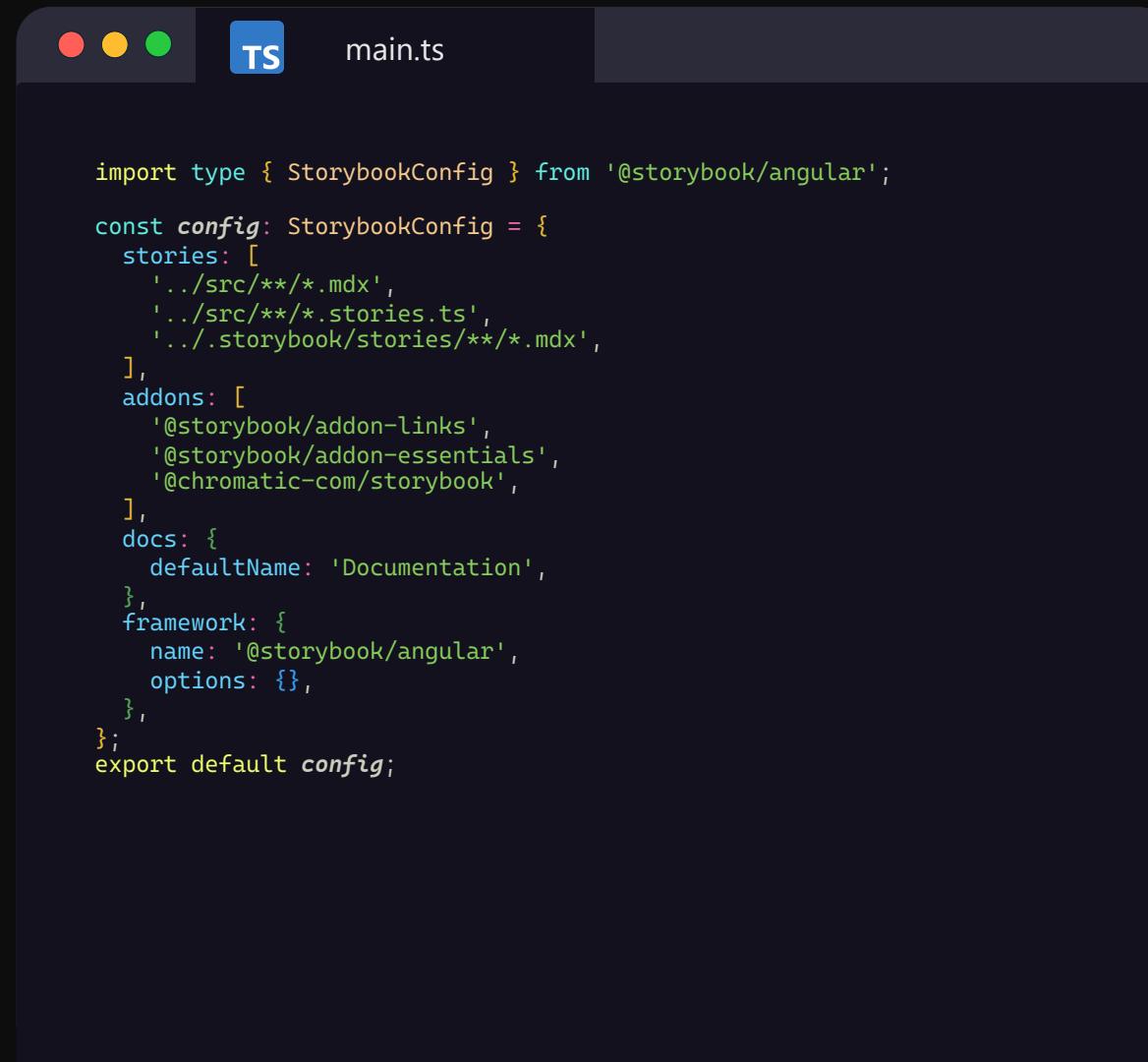
**Page**      **Section**  
**Modal**      **Tab**  
**Overlay**      **Container**

Develop

Document

Test

# StoryBook Configuration



A screenshot of a code editor window titled "main.ts". The window has a dark theme with light-colored text. At the top left are three circular icons (red, yellow, green) and a "TS" icon indicating TypeScript support. The code editor displays the following TypeScript code:

```
import type { StorybookConfig } from '@storybook/angular';

const config: StorybookConfig = {
  stories: [
    '../src/**/*.{mdx,stories.ts}',
    '../storybook/stories/**/*.{mdx,stories.ts}'
  ],
  addons: [
    '@storybook/addon-links',
    '@storybook/addon-essentials',
    '@chromatic/com/storybook'
  ],
  docs: {
    defaultName: 'Documentation'
  },
  framework: {
    name: '@storybook/angular',
    options: {}
  }
};
export default config;
```



TS

main.ts

```
import type { StorybookConfig } from '@storybook/angular';

const config: StorybookConfig = {
  stories: [
    './src/**/*.{mdx,stories.ts}',
    './storybook/stories/**/*.{mdx,stories.ts}'
  ],
  addons: [
    '@storybook/addon-links',
    '@storybook/addon-essentials',
    '@chromatic/com/storybook'
  ],
  docs: {
    defaultName: 'Documentation'
  },
  framework: {
    name: '@storybook/angular',
    options: {}
  }
};
export default config;
```



MDX

angular-guidelines.mdx

```
import { Meta } from '@storybook/blocks';

<Meta title="Guides/Angular" />

# Angular Guidelines

Here you will find guidelines for writing Angular components and services.

## Components

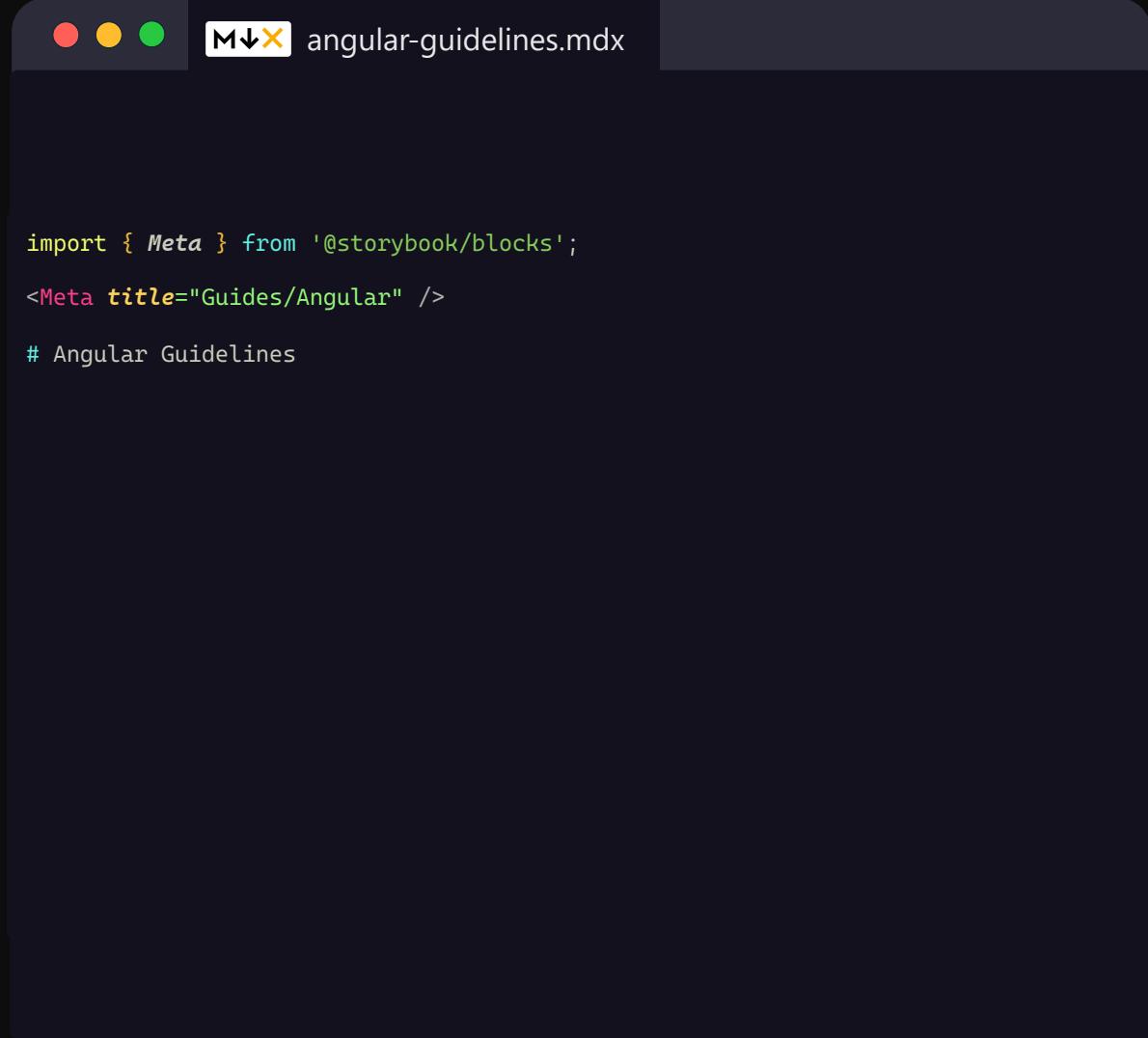
### Do not use services with state in presentational components

Presentational components should not have any dependencies on services. They should only receive data through inputs and emit events through outputs.

#### Do Not

```ts
export class ButtonComponent {
  state: inject(StateService)
}
```

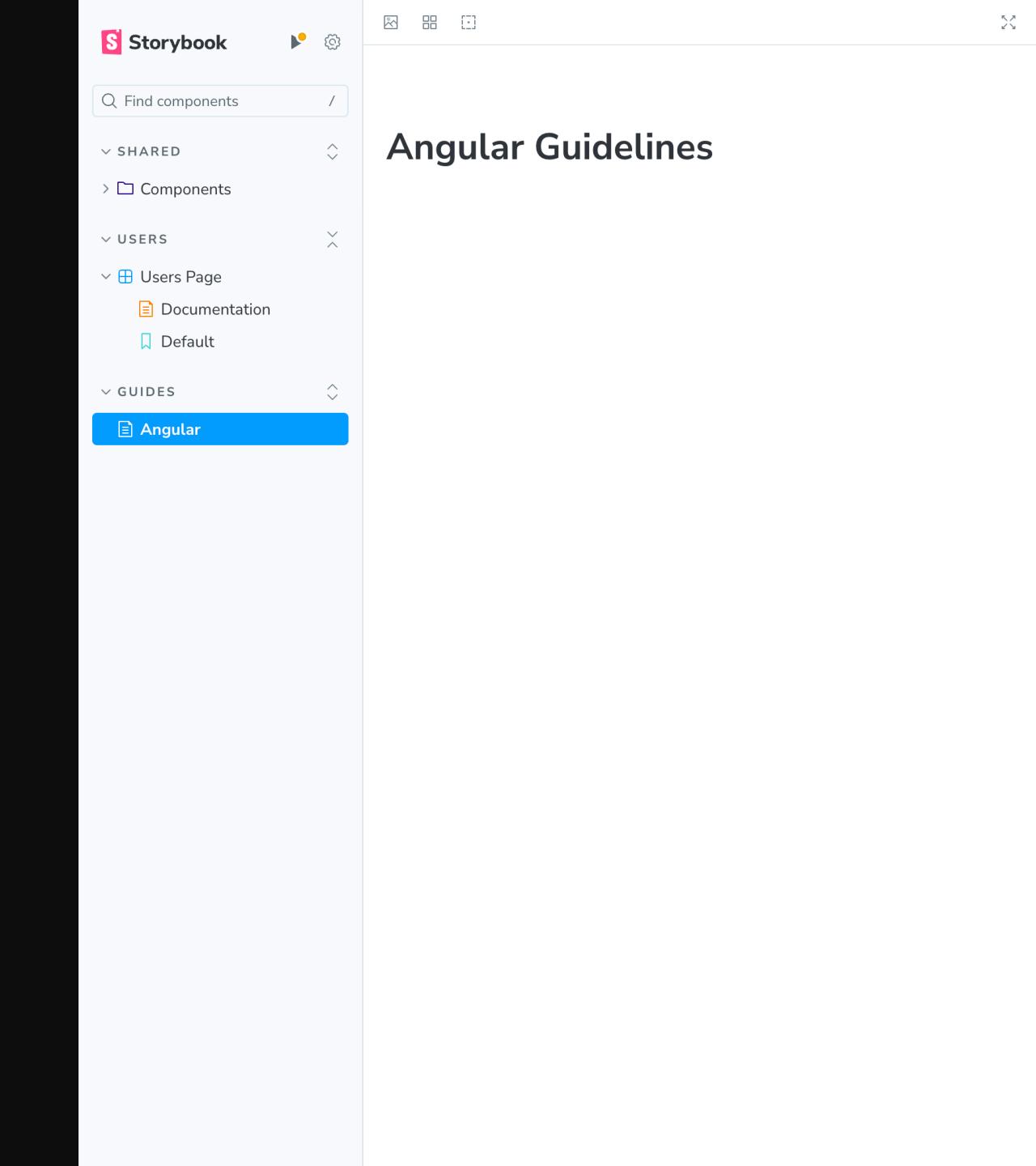
```



```
import { Meta } from '@storybook/blocks';

<Meta title="Guides/Angular" />

# Angular Guidelines
```



Storybook

Find components /

SHARED

Components

USERS

Users Page

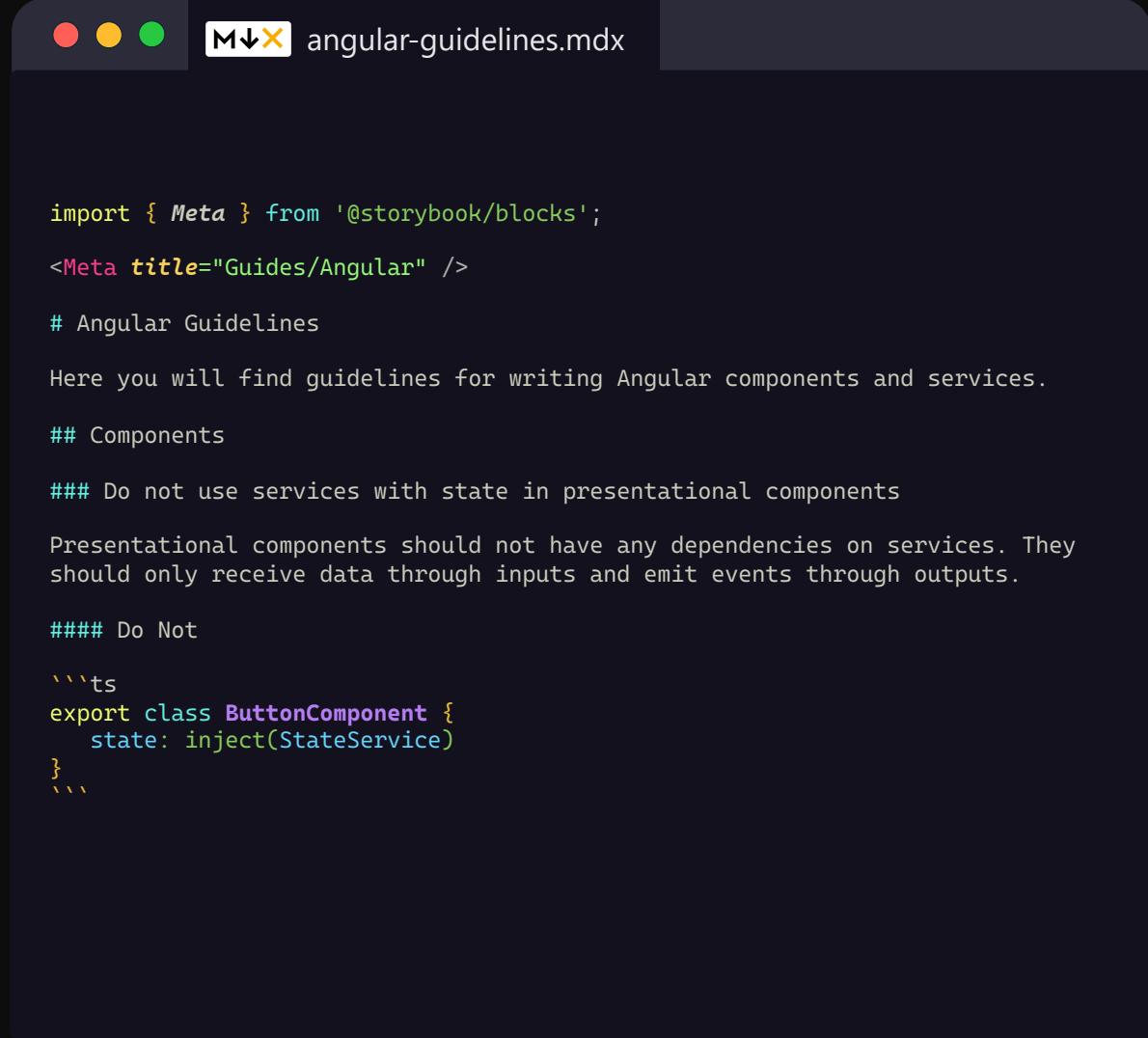
Documentation

Default

GUIDES

Angular

# Angular Guidelines



```
import { Meta } from '@storybook/blocks';

<Meta title="Guides/Angular" />

# Angular Guidelines

Here you will find guidelines for writing Angular components and services.

## Components

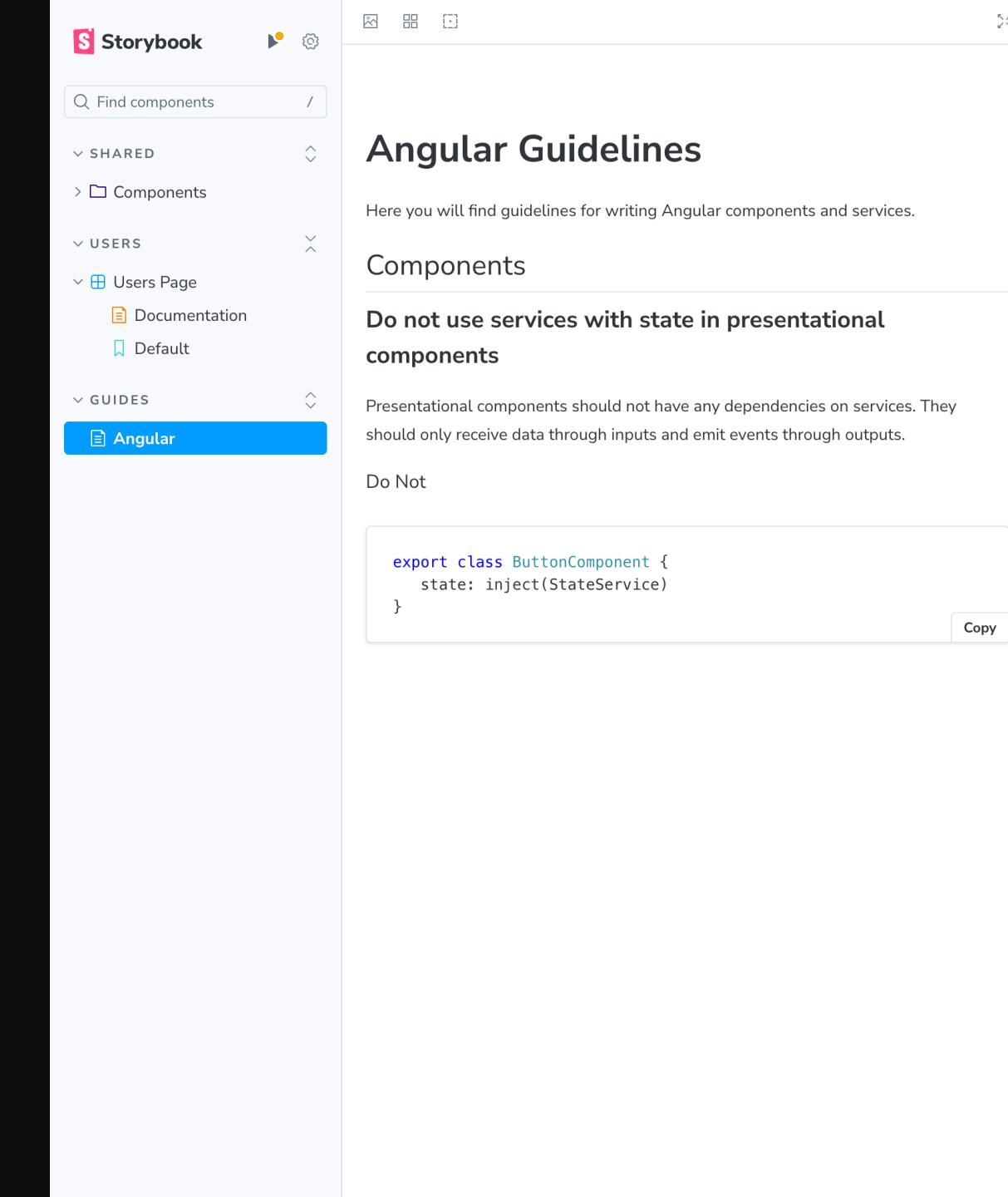
### Do not use services with state in presentational components

Presentational components should not have any dependencies on services. They should only receive data through inputs and emit events through outputs.

#### Do Not

```ts
export class ButtonComponent {
  state: inject(StateService)
}
```

```



**Angular Guidelines**

Here you will find guidelines for writing Angular components and services.

## Components

**Do not use services with state in presentational components**

Presentational components should not have any dependencies on services. They should only receive data through inputs and emit events through outputs.

Do Not

```
export class ButtonComponent {
  state: inject(StateService)
}
```

**Copy**

## **Exercise**

# **Creating a Guideline**

- Create a guideline file under `./storybook/stories`
- Set the location of the story to show on `Guides`
- Configure StoryBook to show the guides first

5:00



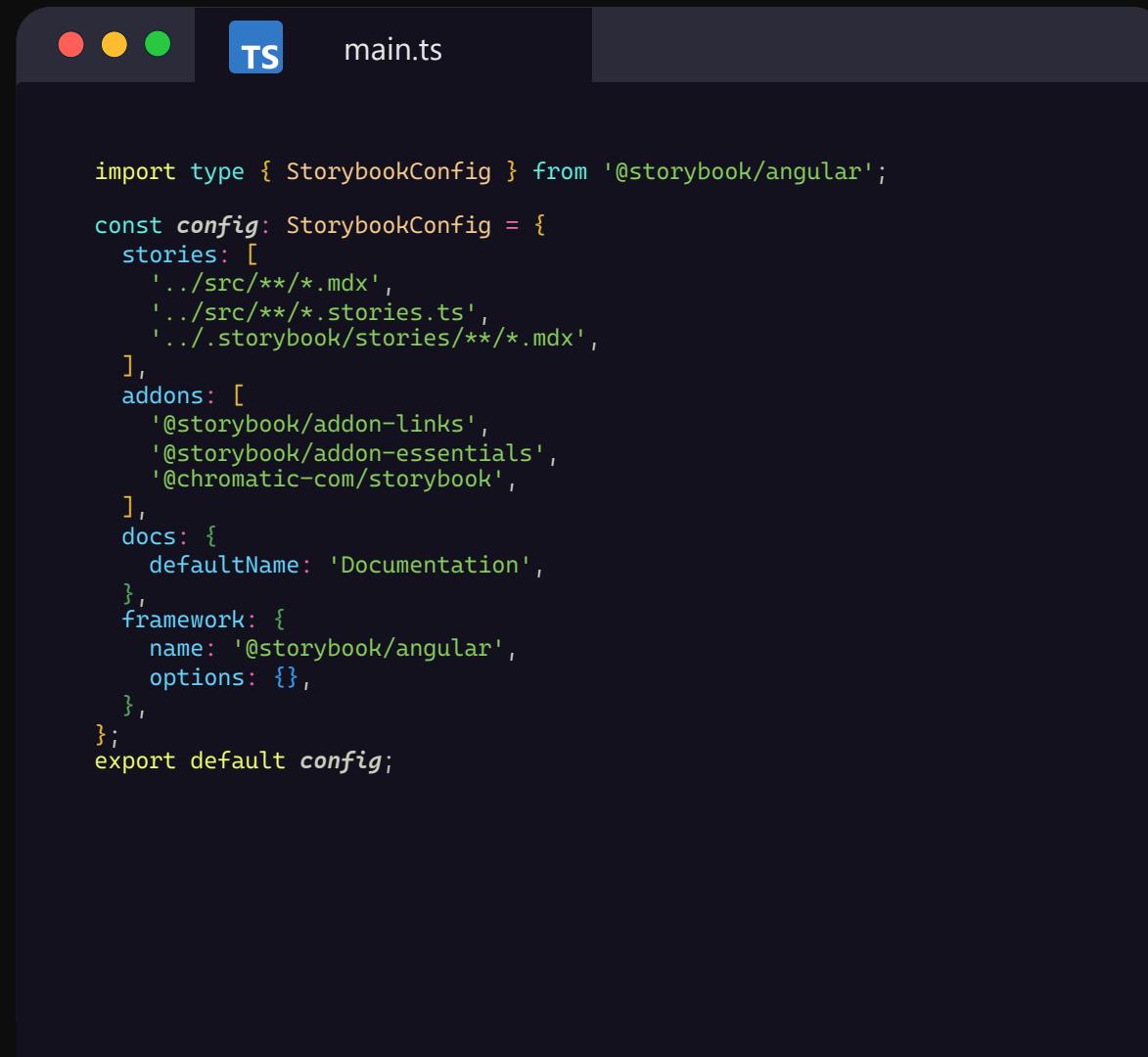
3-creating-a-guideline

Develop

Document

Test

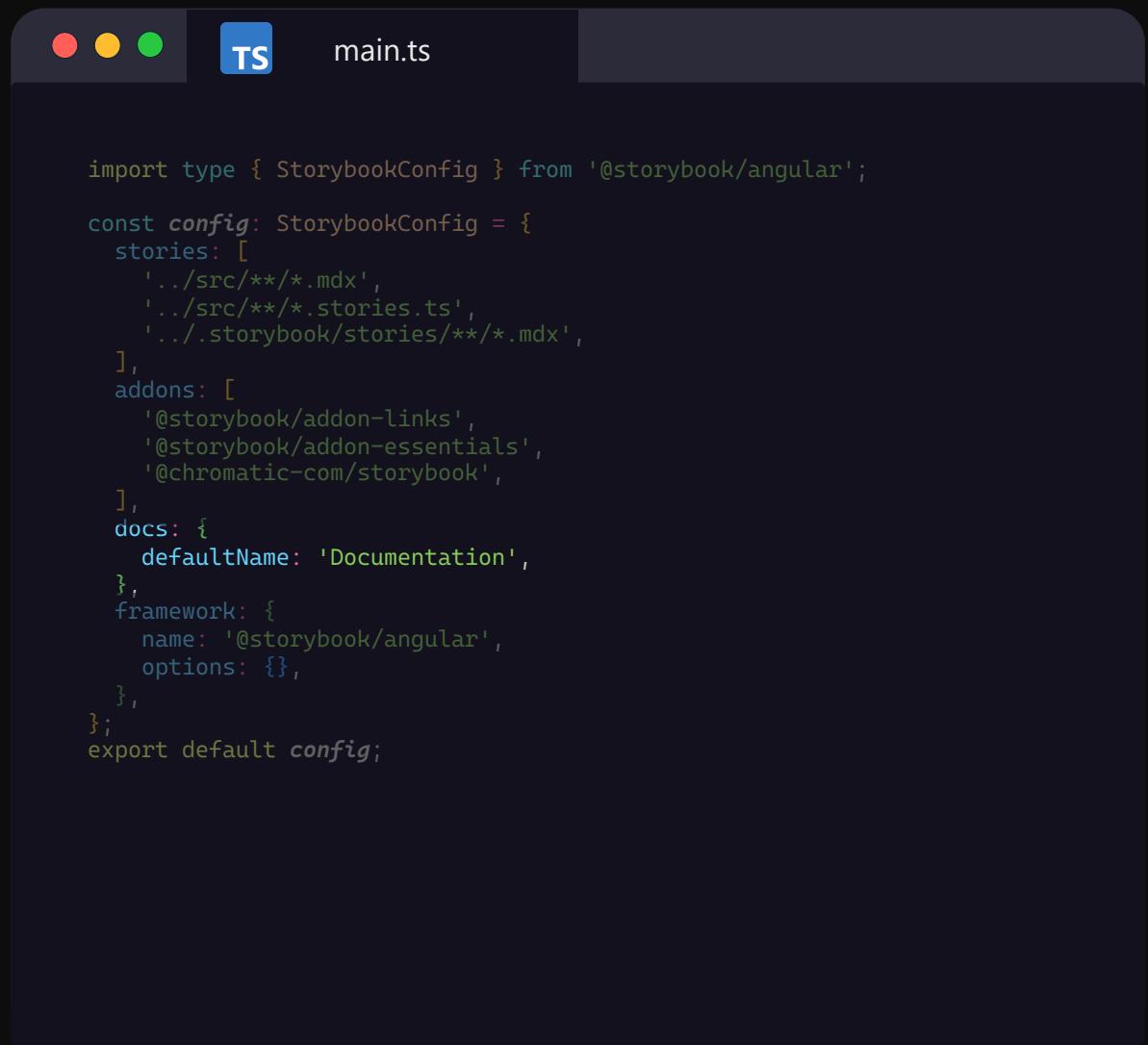
# StoryBook Configuration



A screenshot of a code editor window titled "main.ts". The window has a dark theme with light-colored text. At the top left are three circular icons (red, yellow, green) and a "TS" icon indicating TypeScript support. The code editor displays the following TypeScript code:

```
import type { StorybookConfig } from '@storybook/angular';

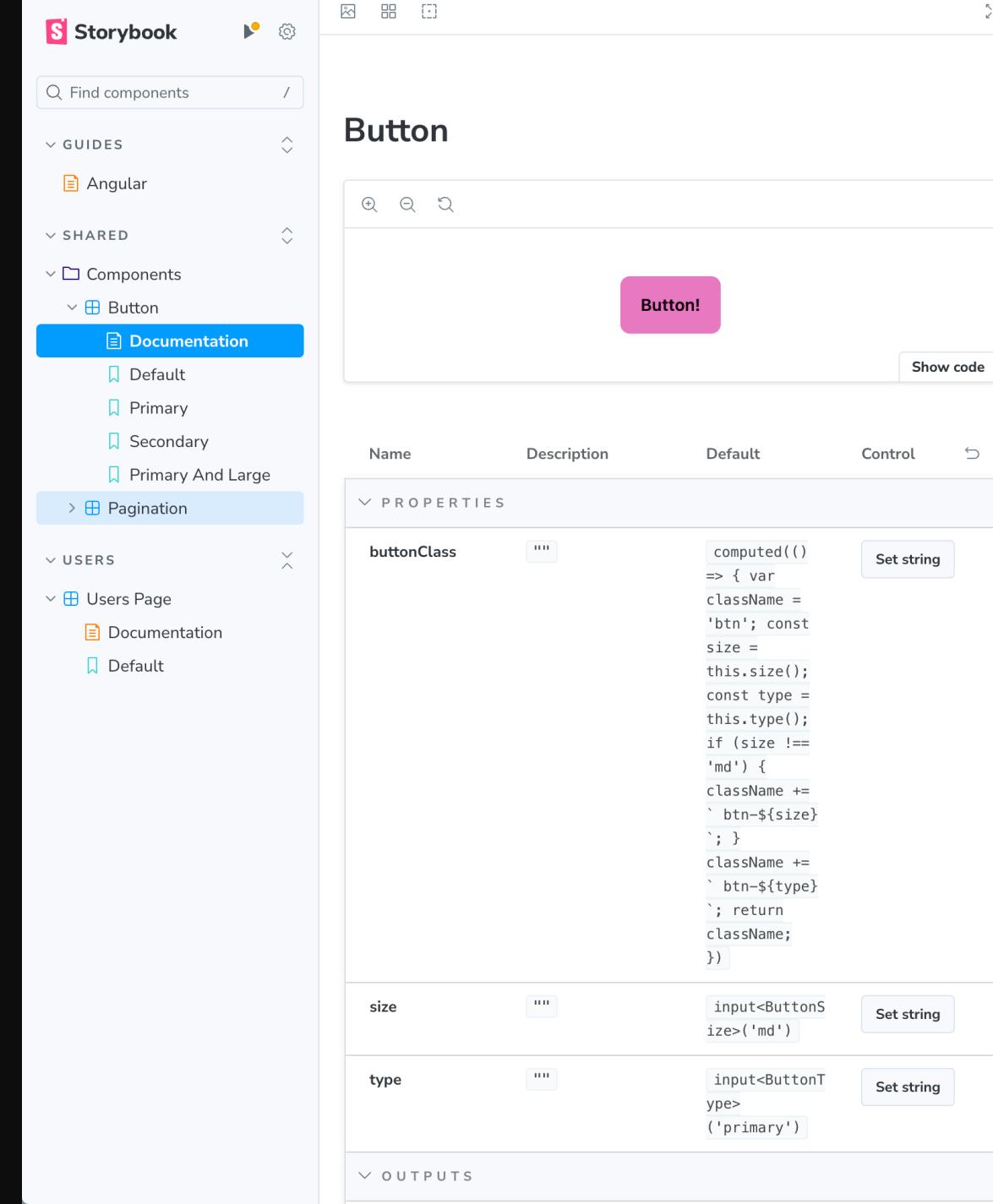
const config: StorybookConfig = {
  stories: [
    '../src/**/*.{mdx,stories.ts}',
    '../storybook/stories/**/*.{mdx,stories.ts}'
  ],
  addons: [
    '@storybook/addon-links',
    '@storybook/addon-essentials',
    '@chromatic/com/storybook'
  ],
  docs: {
    defaultName: 'Documentation'
  },
  framework: {
    name: '@storybook/angular',
    options: {}
  }
};
export default config;
```



main.ts

```
import type { StorybookConfig } from '@storybook/angular';

const config: StorybookConfig = {
  stories: [
    './src/**/*.{mdx,stories.ts}',
    './storybook/stories/**/*.{mdx,stories.ts}'
  ],
  addons: [
    '@storybook/addon-links',
    '@storybook/addon-essentials',
    '@chromatic/com/storybook'
  ],
  docs: {
    defaultName: 'Documentation',
  },
  framework: {
    name: '@storybook/angular',
    options: {},
  },
};
export default config;
```



Storybook

Find components /

GUIDES

- Angular

SHARED

Components

- Button
- Documentation
- Default
- Primary
- Secondary
- Primary And Large

Pagination

USERS

Users Page

- Documentation
- Default

## Button

Button!

Show code

| Name        | Description   | Default    | Control |
|-------------|---|------------|---------|
| buttonClass | computed() => { var className = 'btn'; const size = this.size(); const type = this.type(); if (size !== 'md') { className += ` btn-\${size}`; } className += ` btn-\${type}`; return className; } | Set string |         |
| size        | input<ButtonType>('md')   | Set string |         |
| type        | input<ButtonType>('primary')  | Set string |         |

```

import { Meta, StoryObj, argsToTemplate } from '@storybook/angular';
import { ButtonComponent } from './button.component';

const meta: Meta<ButtonComponent> = {
  title: 'Shared/Components/Button',
  component: ButtonComponent,
  tags: ['!autodocs'],
  render: (args) => ({
    props: args,
    template: `<ui-button ${argsToTemplate(args)}>Button!</ui-button>`,
  }),
};

export default meta;
type Story = StoryObj<ButtonComponent>;

export const Default: Story = {};

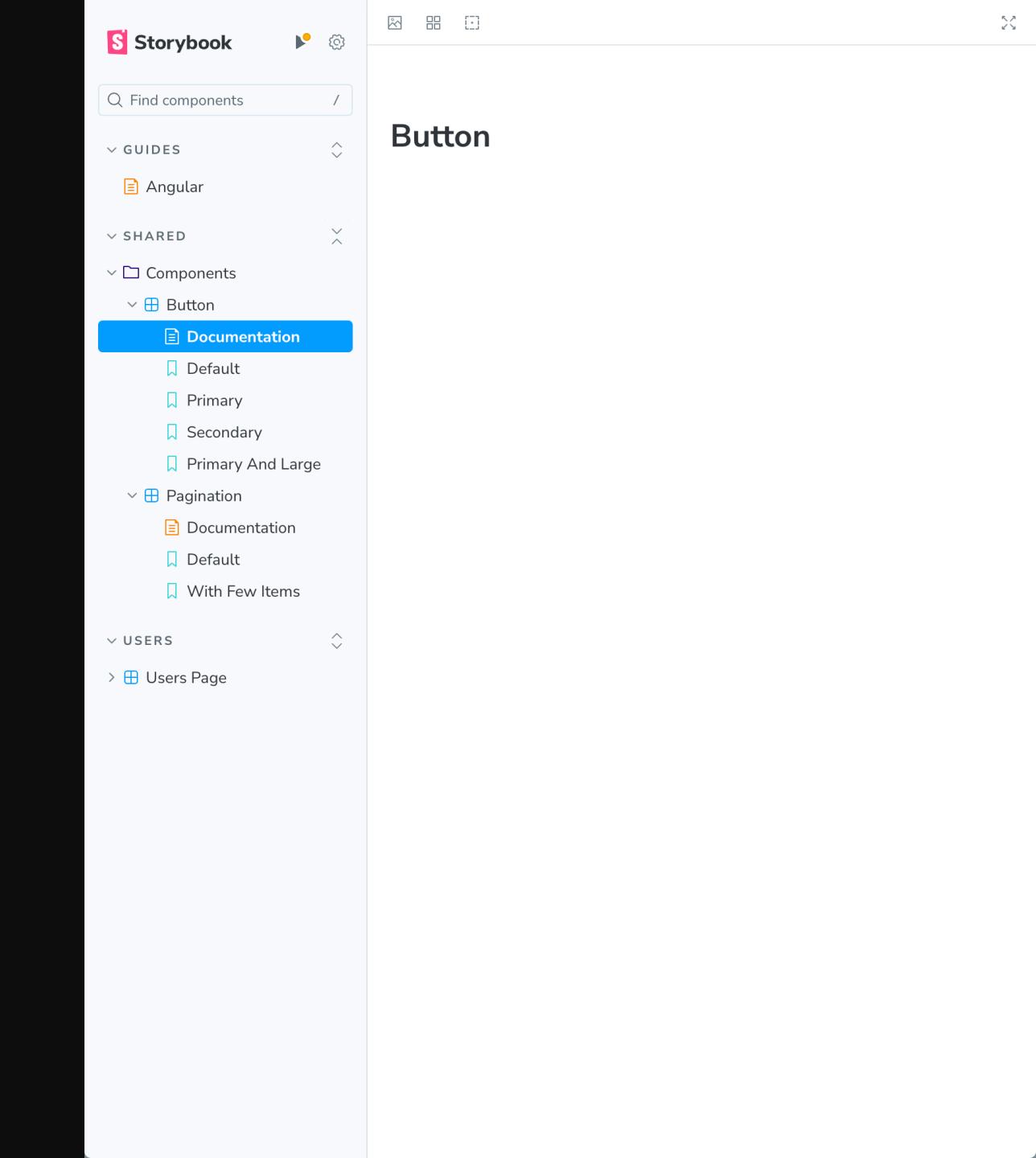
export const Primary = {
  args: {
    size: 'md',
    type: 'primary',
  },
};

export const Secondary = {
  args: {
    type: 'secondary',
  },
};

export const PrimaryAndLarge = {
  args: {
    size: 'lg',
    type: 'primary',
  },
};

```

| Name        | Description  | Default    | Control |
|-------------|--|------------|---------|
| buttonClass | computed() => { var className = 'btn'; const size = this.size(); const type = this.type(); if (size !== 'md') { className += ` btn-\${size}`; } className += ` btn-\${type}`; return className; }) | Set string |         |
| size        | input<ButtonType>('md')  | Set string |         |
| type        | input<ButtonType>('primary')   | Set string |         |



```
import { Controls, Meta, Story, Title } from '@storybook/blocks';
import * as ButtonStories from './button.stories';
```

## # Description

The button component is a simple button that can be used to trigger actions in the application. It can be used to submit forms, navigate to a different page, or trigger any other action that requires user input.

## # Behavior

Clicking the button will trigger the `'click'` event. This event can be used to trigger actions in the application.

## Primary

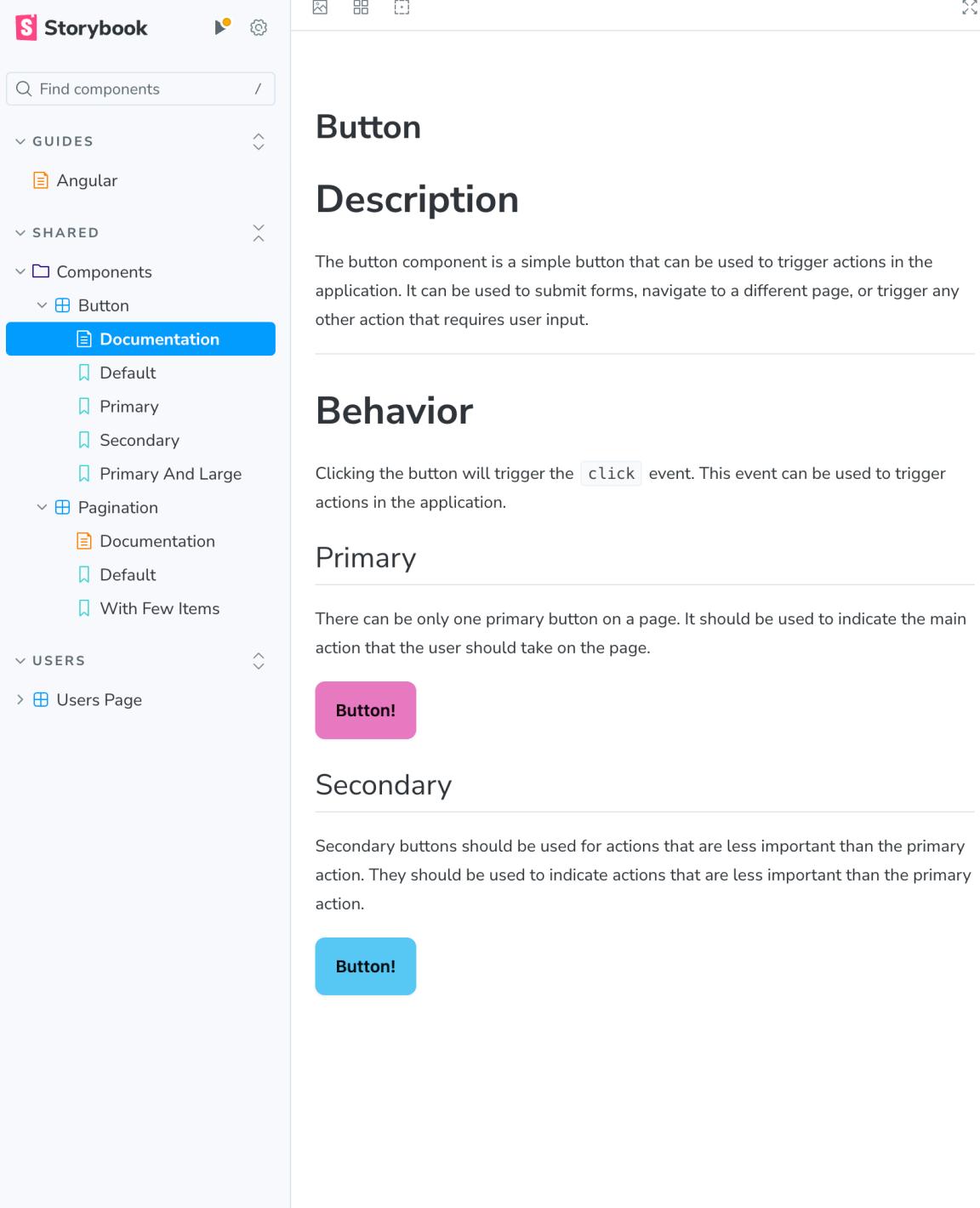
There can be only one primary button on a page. It should be used to indicate the main action that the user should take on the page.

```
<Story of={ButtonStories.Primary} />
```

## ## Secondary

Secondary buttons should be used for actions that are less important than the primary action. They should be used to indicate actions that are less important than the primary action.

```
<Story of={ButtonStories.Secondary} />
```



Button.mdx

```

import { Controls, Meta, Story, Title } from '@storybook/blocks';
import * as ButtonStories from './button.stories';

<Meta of={ButtonStories} />
<Title of={ButtonStories} />

# Description

The button component is a simple button that can be used to trigger actions in the application. It can be used to submit forms, navigate to a different page, or trigger any other action that requires user input.

---

# Behavior

Clicking the button will trigger the `click` event. This event can be used to trigger actions in the application.

## Primary

There can be only one primary button on a page. It should be used to indicate the main action that the user should take on the page.

<Story of={ButtonStories.Primary} />

## Secondary

Secondary buttons should be used for actions that are less important than the primary action. They should be used to indicate actions that are less important than the primary action.

<Story of={ButtonStories.Secondary} />

---

# Inputs

<Controls />

```

**Storybook**

Find components /

- GUIDES
  - Angular
- SHARED
- Components
  - Button
  - Documentation
  - Default
  - Primary
  - Secondary
  - Primary And Large
- Pagination
  - Documentation
  - Default
  - With Few Items
- USERS
  - Users Page

## Primary

There can be only one primary button on a page. It should be used to indicate the main action that the user should take on the page.

**Button!**

## Secondary

Secondary buttons should be used for actions that are less important than the primary action. They should be used to indicate actions that are less important than the primary action.

**Button!**

## Inputs

Name	Description	Default	Control
PROPERTIES			
buttonClass	... computed(() => { var className = 'btn'; const size = this.size(); const type = this.type(); if (size !== 'md') { className += ` btn-\${size}`; } className += ` btn-\${type}`; return className; })	Set string	
size	... input<Buttons>	String	

## **Exercise**

# Creating a custom documentation

- Disable the documentation generated in the pagination
- Create documentation file (`pagination.mdx`) for the pagination Component
- Make sure to Use `Meta`, `Title` and `Controls`
- Add sections for Description and Behavior
- Showcase two of the states (stories) using the `<Story of={} />` from '@storybook/blocks'
- Add a section for the inputs

5:00



4-custom-component-documentation

Develop

Document

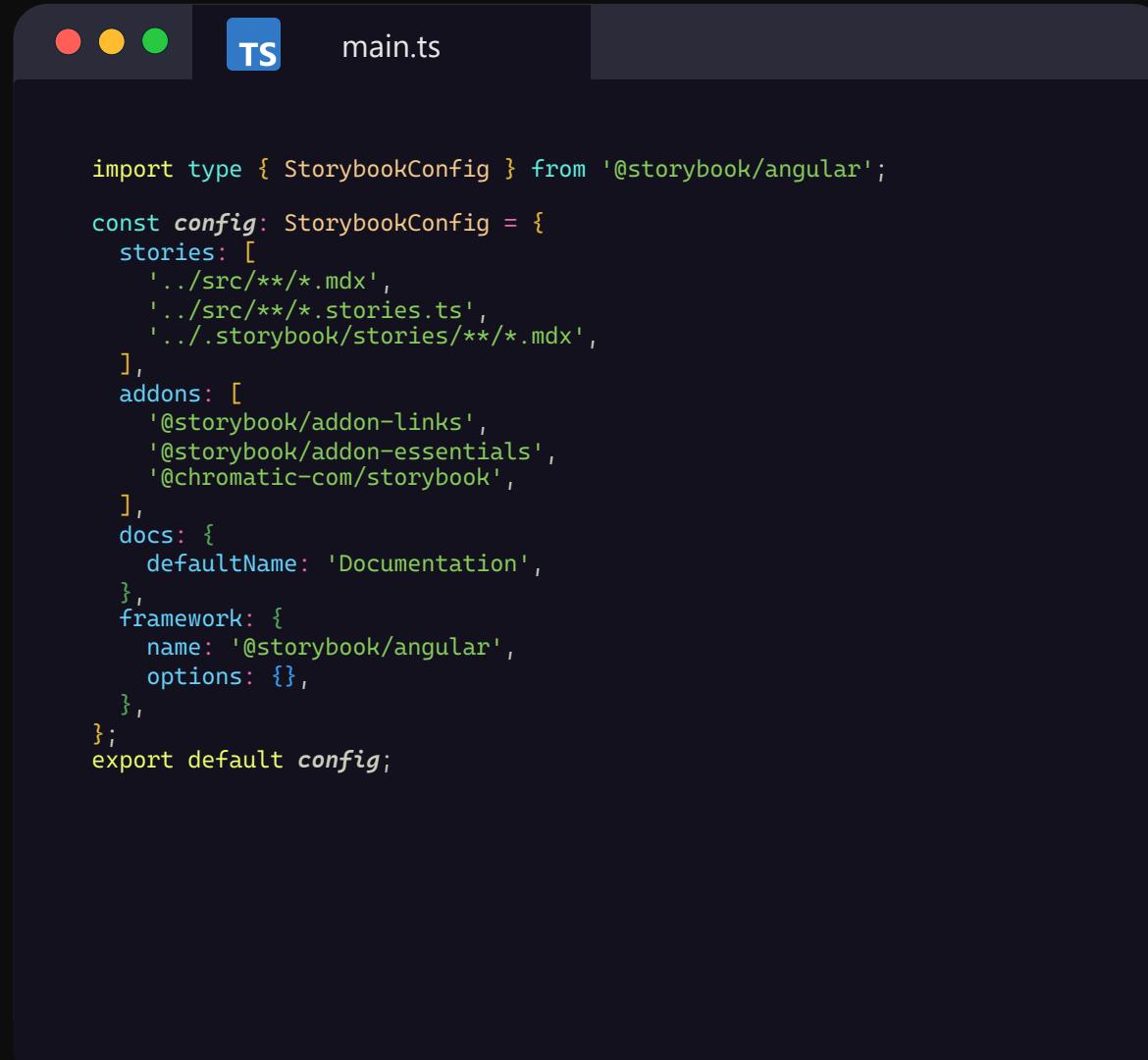
Test

Test

Accessibility

Interactions

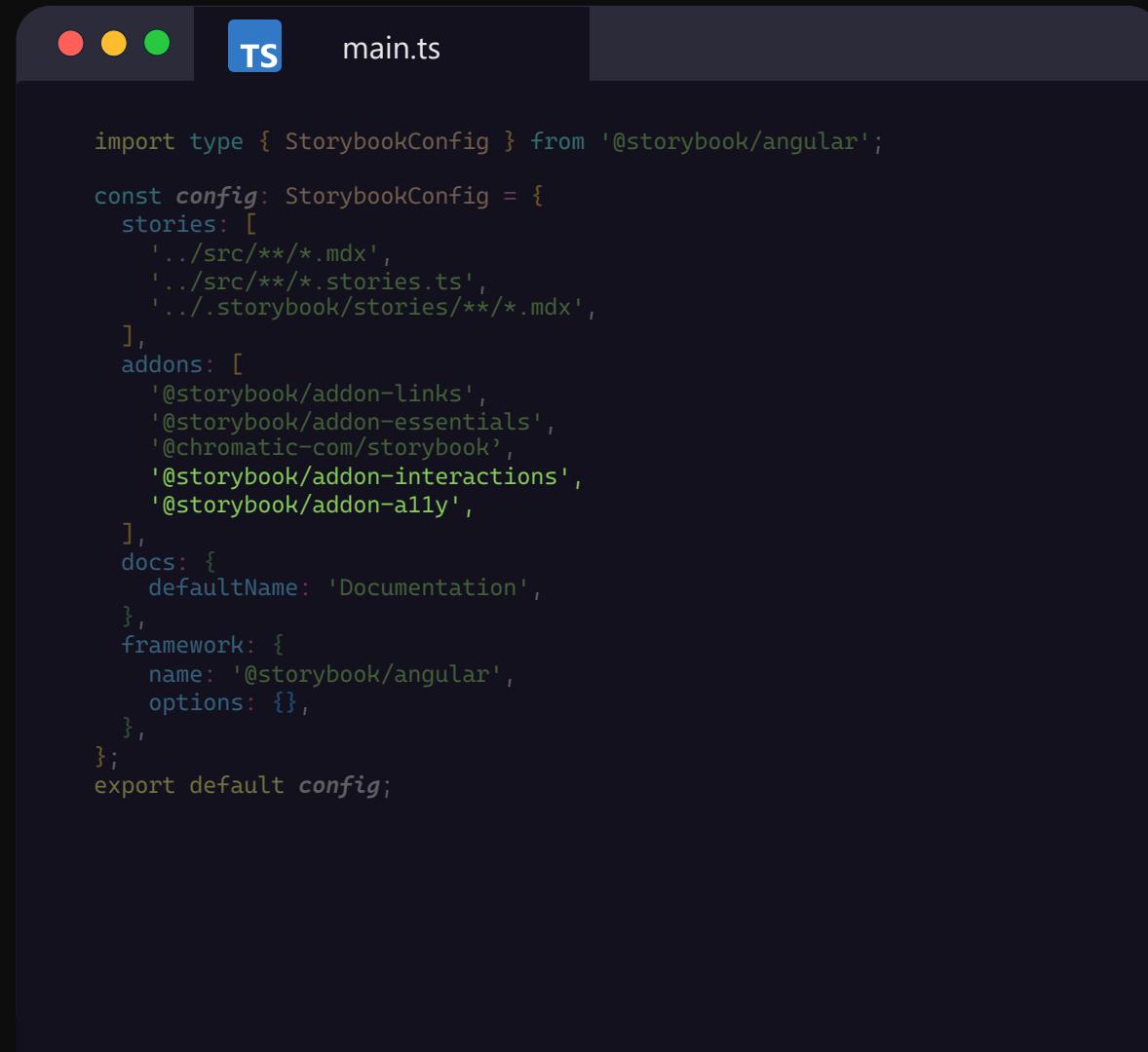
# Addons



The image shows a screenshot of a code editor window with a dark theme. The title bar says "main.ts". The code editor displays the following TypeScript code:

```
import type { StorybookConfig } from '@storybook/angular';

const config: StorybookConfig = {
  stories: [
    '../src/**/*.{mdx,stories.ts}',
    '../storybook/stories/**/*.{mdx}'
  ],
  addons: [
    '@storybook/addon-links',
    '@storybook/addon-essentials',
    '@chromatic/com/storybook'
  ],
  docs: {
    defaultName: 'Documentation'
  },
  framework: {
    name: '@storybook/angular',
    options: {}
  }
};
export default config;
```



A screenshot of a dark-themed code editor window titled "main.ts". The window includes standard OS X-style window controls (red, yellow, green) and a "TS" icon indicating TypeScript support. The code editor displays the following TypeScript configuration for Storybook:

```
import type { StorybookConfig } from '@storybook/angular';

const config: StorybookConfig = {
  stories: [
    '../src/**/*.{mdx,stories.ts}',
    './storybook/stories/**/*.{mdx}'
  ],
  addons: [
    '@storybook/addon-links',
    '@storybook/addon-essentials',
    '@chromatic/com/storybook',
    '@storybook/addon-interactions',
    '@storybook/addon-a11y'
  ],
  docs: {
    defaultName: 'Documentation'
  },
  framework: {
    name: '@storybook/angular',
    options: {}
  }
};
export default config;
```

The screenshot shows a Mac OS X desktop environment. In the foreground, there is a terminal window with a blue background and a code editor window for a file named 'main.ts'. The code editor has a dark theme with syntax highlighting. The 'main.ts' file contains configuration code for Storybook, including paths for stories and addons, and a default documentation page.

```
import type { StorybookConfig } from '@storybook/angular';

const config: StorybookConfig = {
  stories: [
    '../src/**/*.{mdx,stories.ts}',
    '../storybook/stories/**/*.{mdx}'
  ],
  addons: [
    '@storybook/addon-links',
    '@storybook/addon-essentials',
    '@chromatic/com/storybook',
    '@storybook/addon-interactions',
    '@storybook/addon-ally'
  ],
  docs: {
    defaultName: 'Documentation'
  },
  framework: {
    name: '@storybook/angular',
    options: {}
  }
};
export default config;
```

The screenshot shows the Storybook application interface. The left sidebar contains navigation links for 'GUIDES' (Angular), 'SHARED' (Components, Pagination, Documentation, Default, With Few Items), and 'USERS' (Users Page). The main content area is dark, showing a navigation bar with pages 1 through 10. At the bottom, an 'Accessibility' audit panel is open, showing results for 10 controls. The panel includes sections for 'Buttons must have discernible text', 'Ensures buttons have discernible text', and a detailed list of 6 critical violations related to button accessibility.

Storybook

Find components /

GUIDES

Angular

SHARED

Components

Button

Pagination

Documentation

Default

With Few Items

USERS

Users Page

« 1 2 3 4 5 6 7 8 9 10 »

Controls 10 Actions Visual Tests Interactions Accessibility

0 Violations 3 Passes 0 Incomplete

Buttons must have discernible text

Ensures buttons have discernible text

More info...

1. join-item.btn-secondary.btn:nth-child(1)

Critical Element has inner text that is visible to screen readers

2. .btn-active

Critical Element has inner text that is visible to screen readers

3. join-item.btn-secondary[ng-reflect-ng-class="[object Object]"]:nth-child(3)

Critical Element has inner text that is visible to screen readers

4. join-item.btn-secondary[ng-reflect-ng-class="[object Object]"]:nth-child(4)

Critical Element has inner text that is visible to screen readers

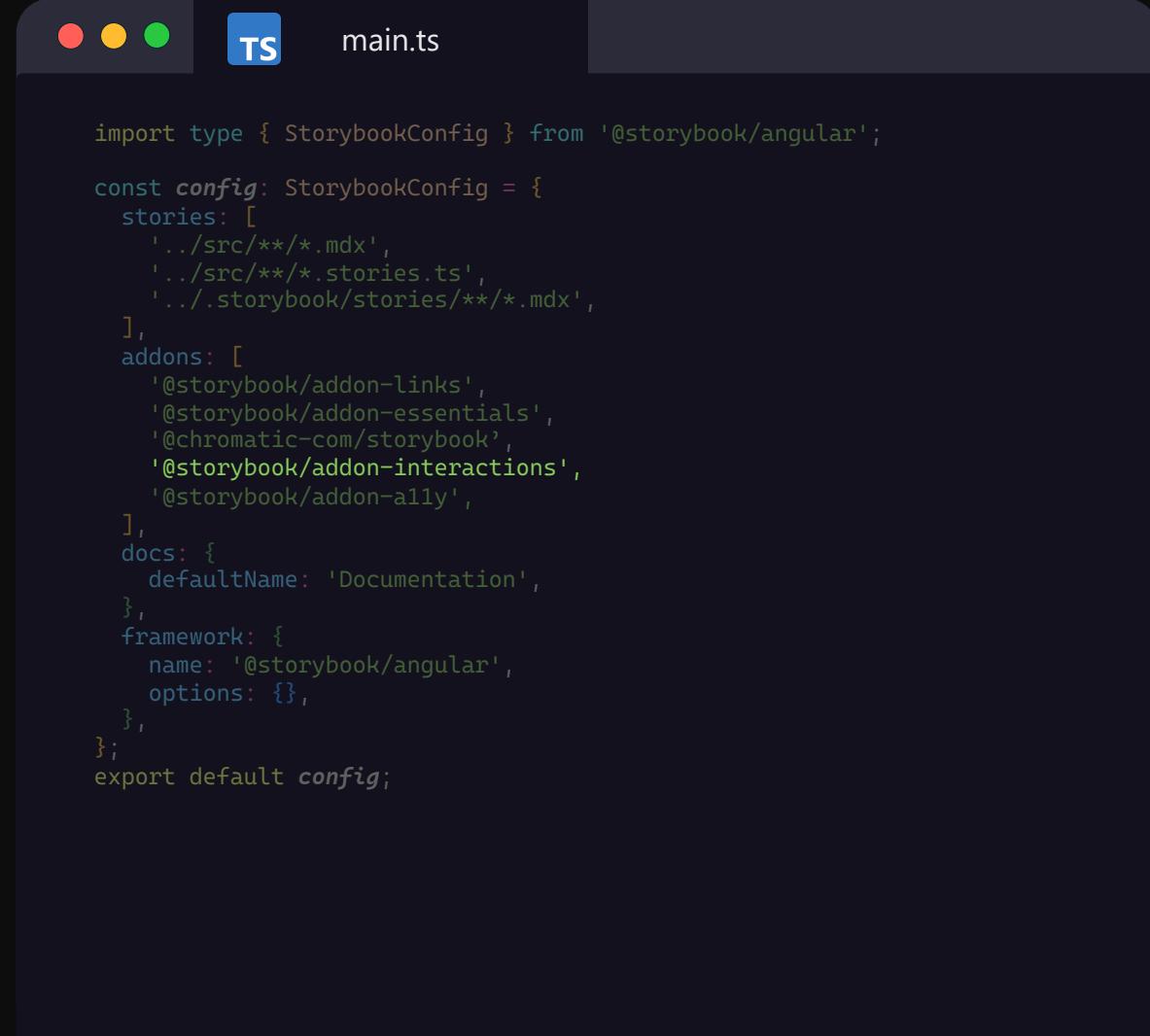
5. join-item.btn-secondary[ng-reflect-ng-class="[object Object]"]:nth-child(5)

Critical Element has inner text that is visible to screen readers

6. join-item.btn-secondary[ng-reflect-ng-class="[object Object]"]:nth-child(6)

Critical Element has inner text that is visible to screen readers

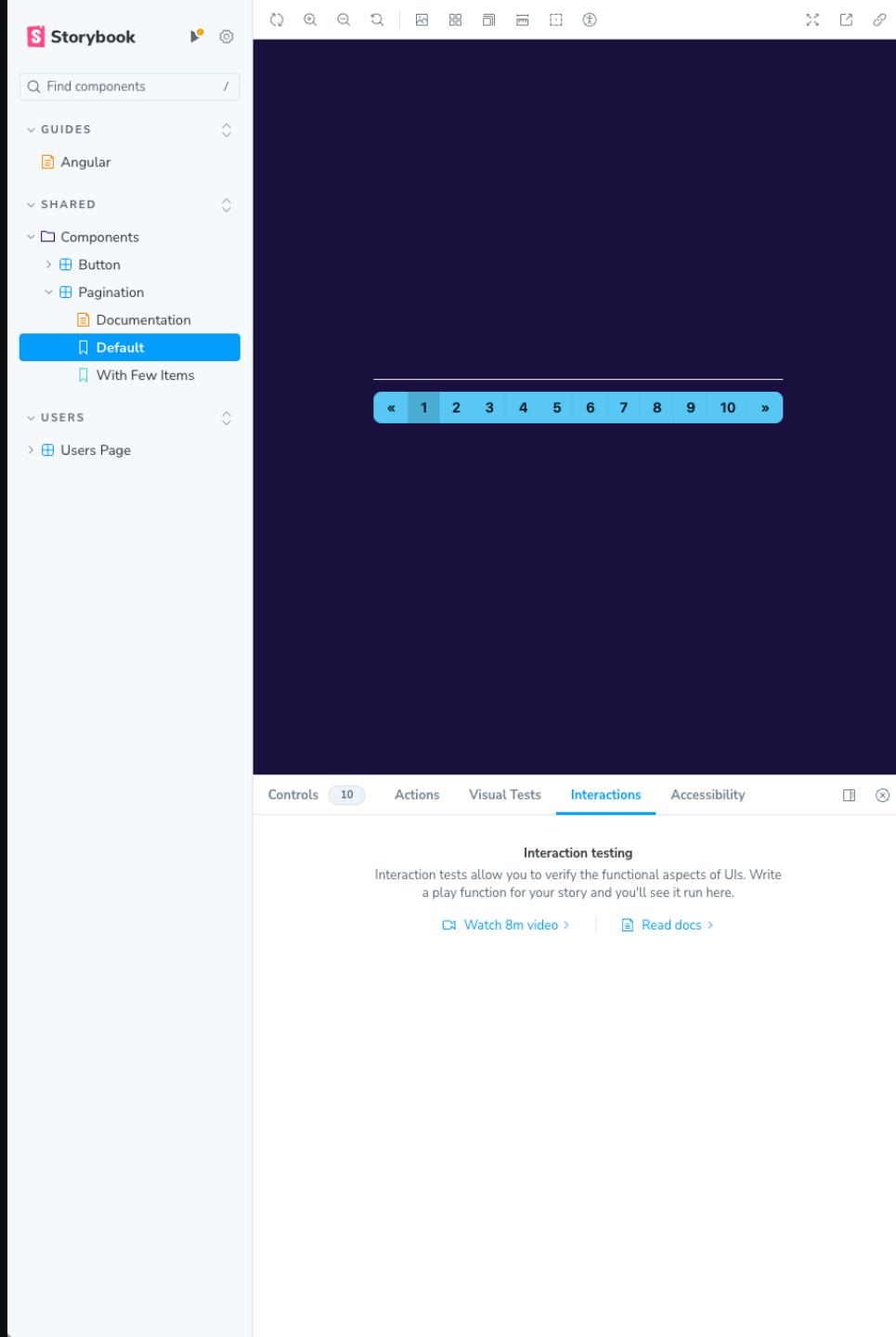
Tests completed



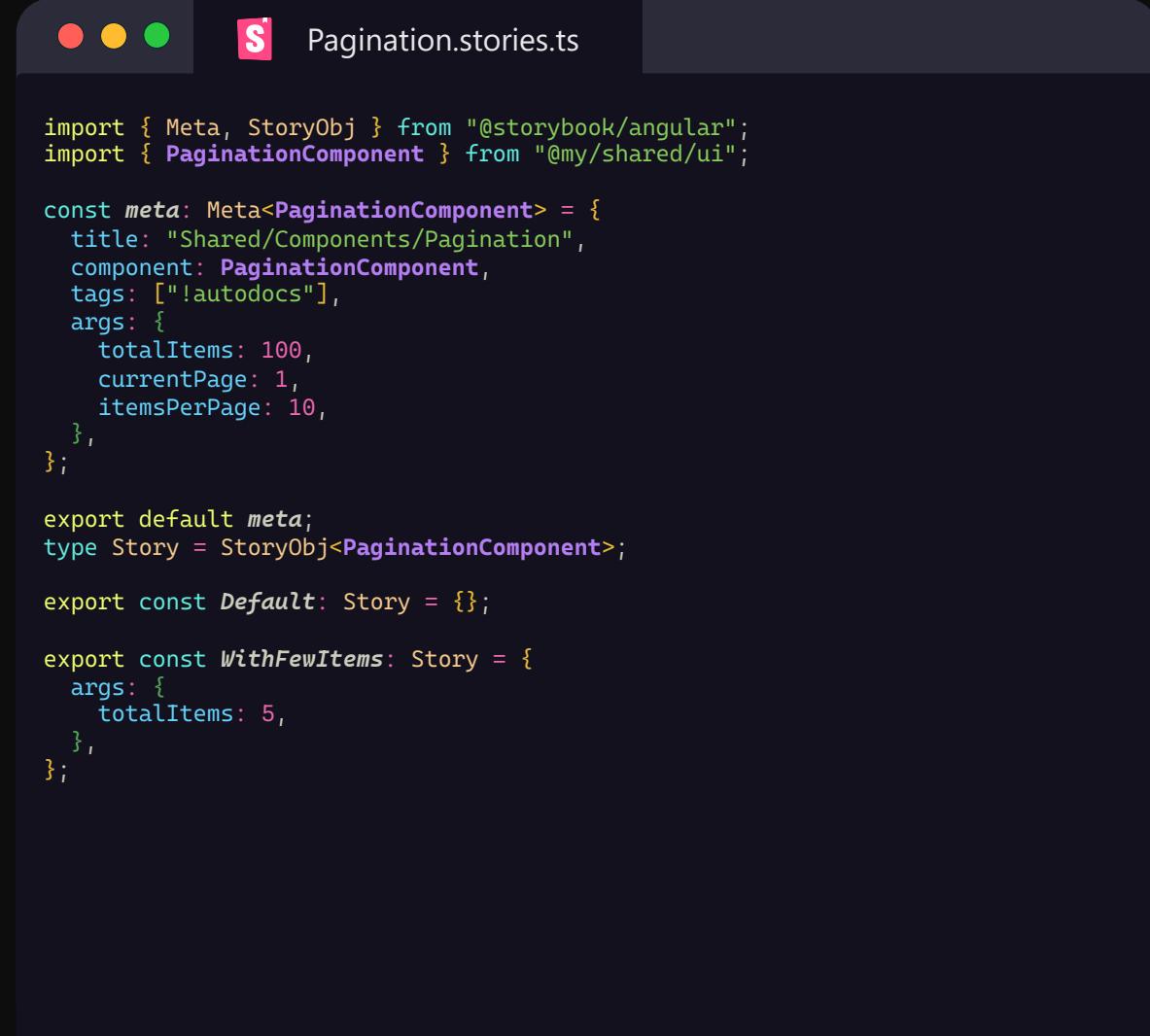
A screenshot of a code editor window titled "main.ts". The code editor displays the following TypeScript configuration:

```
import type { StorybookConfig } from '@storybook/angular';

const config: StorybookConfig = {
  stories: [
    '../src/**/*.{mdx,stories.ts}',
    '../storybook/stories/**/*.{mdx}'
  ],
  addons: [
    '@storybook/addon-links',
    '@storybook/addon-essentials',
    '@chromatic/com/storybook',
    '@storybook/addon-interactions',
    '@storybook/addon-ally'
  ],
  docs: {
    defaultName: 'Documentation'
  },
  framework: {
    name: '@storybook/angular',
    options: {}
  }
};
export default config;
```



A screenshot of the Storybook interface. The left sidebar shows a tree view of components under "Components": "Angular", "Button", "Pagination", "Documentation", and "Default" (which is selected). Below this are sections for "GUIDES", "SHARED", and "USERS". The right panel displays the "Default" component with a preview area showing a dark blue card with a white border and some placeholder text. At the bottom, there are tabs for "Controls", "Actions", "Visual Tests", "Interactions" (which is active), and "Accessibility". A callout for "Interaction testing" explains that it allows verifying functional aspects of UIs. Navigation controls at the bottom show page 10 of 10.



A screenshot of a code editor window titled "Pagination.stories.ts". The code is written in TypeScript and defines two stories for a "PaginationComponent". The first story, "Default", uses default arguments: totalItems: 100, currentPage: 1, and itemsPerPage: 10. The second story, "With Few Items", uses arguments: totalItems: 5. Both stories import "Meta" and "StoryObj" from "@storybook/angular" and "PaginationComponent" from "@my/shared/ui".

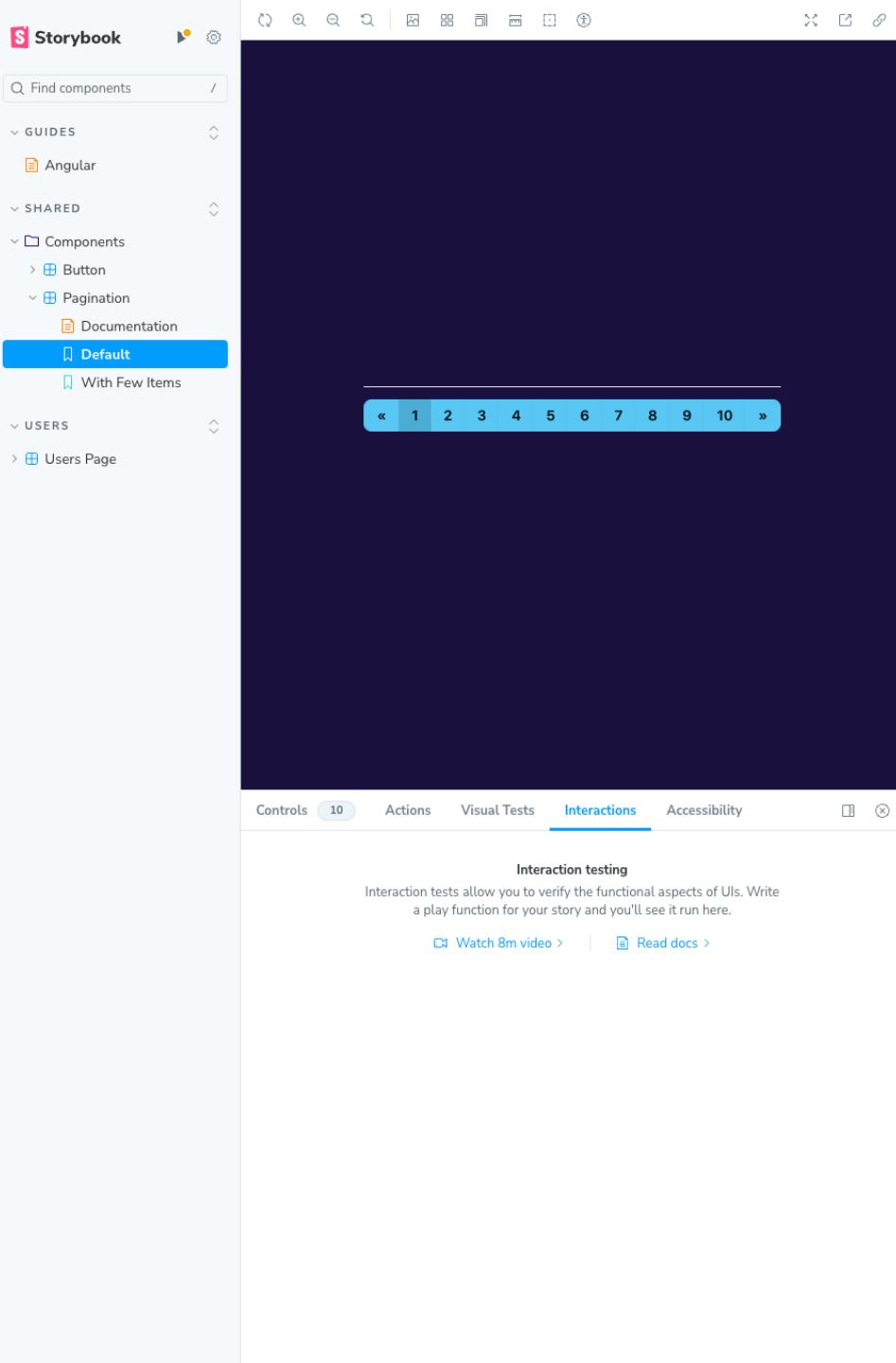
```
import { Meta, StoryObj } from "@storybook/angular";
import { PaginationComponent } from "@my/shared/ui";

const meta: Meta<PaginationComponent> = {
  title: "Shared/Components/Pagination",
  component: PaginationComponent,
  tags: ["!autodocs"],
  args: {
    totalItems: 100,
    currentPage: 1,
    itemsPerPage: 10,
  },
};

export default meta;
type Story = StoryObj<PaginationComponent>;

export const Default: Story = {};

export const WithFewItems: Story = {
  args: {
    totalItems: 5,
  },
};
```



A screenshot of the Storybook interface. The left sidebar shows a tree structure of components: GUIDES (Angular), SHARED (Components: Button, Pagination - selected, Documentation, Default - selected, With Few Items), and USERS (Users Page). The right panel displays the "Default" pagination story, which shows a page with 10 items per page, currently on page 1 of 10. Below the preview, there are tabs for Controls, Actions, Visual Tests, Interactions (which is selected), and Accessibility. A note about "Interaction testing" is present, along with links to a video and documentation.

Storybook

Find components /

GUIDES

Angular

SHARED

Components

Button

Pagination

Documentation

Default

With Few Items

USERS

Users Page

« 1 2 3 4 5 6 7 8 9 10 »

Controls 10 Actions Visual Tests **Interactions** Accessibility

Interaction testing

Interaction tests allow you to verify the functional aspects of UIs. Write a play function for your story and you'll see it run here.

[Watch 8m video >](#) | [Read docs >](#)



# Pagination.stories.ts

```
import { Meta, StoryObj } from "@storybook/angular";
import { PaginationComponent } from "@my/shared/ui";
import { userEvent, within, expect } from '@storybook/test';

const meta: Meta<PaginationComponent> = {
  title: "Shared/Components/Pagination",
  component: PaginationComponent,
  tags: ["!autodocs"],
  args: {
    totalItems: 100,
    currentPage: 1,
    itemsPerPage: 10,
  },
};

export default meta;
type Story = StoryObj<PaginationComponent>;

export const Default: Story = {
  play: async ({ canvasElement }) => {
    const canvas = within(canvasElement);

    await expect(canvas.getByText(/100/i)).toBeInTheDocument();
    await expect(canvas.getByText(/results/i)).toBeInTheDocument();
  },
};

export const WithFewItems: Story = {
  args: {
    totalItems: 5,
  },
};
```

The screenshot shows the Storybook interface with the following details:

- Header:** Storybook logo, search bar, and various icons.
- Sidebar:** A navigation tree on the left:
  - GUIDES
  - Angular
  - SHARED
    - Components
      - Button
      - Pagination
        - Documentation
    - Default
    - With Few Items
  - USERS
    - Users Page
- Content Area:** A large dark blue placeholder area representing the component's visual output.
- Page Number:** A navigation bar at the bottom with page numbers from 1 to 10.
- Bottom Bar:** A footer with tabs for Controls (10), Actions, Visual Tests, Interactions (2), Accessibility, and a file icon.
- Test Results:** A table showing test results for the 'pagination.stories.ts' file:

Controls	Actions	Visual Tests	Interactions	Accessibility
<span>PASS</span>	Scroll to end	▶◀ ▶▶ ⏪ ⏩	2	0
<pre>✓ expect(within(&lt;div#storybook-root&gt;).getByText(/100/i)).toBeInTheDocument() ✓ expect(within(&lt;div#storybook-root&gt;).getByText(/results/i)).toBeInTheDocument()</pre>				

Pagination.stories.ts

```
import { Meta, StoryObj } from "@storybook/angular";
import { PaginationComponent } from "@my/shared/ui";
import { userEvent, within, expect } from '@storybook/test';

const meta: Meta<PaginationComponent> = {
  title: "Shared/Components/Pagination",
  component: PaginationComponent,
  tags: ["!autodocs"],
  args: {
    totalItems: 100,
    currentPage: 1,
    itemsPerPage: 10,
  },
};

export default meta;
type Story = StoryObj<PaginationComponent>;

export const Default: Story = {};

export const WithFewItems: Story = {
  args: {
    totalItems: 25,
    itemsPerPage: 5,
  },
  play: async ({ canvasElement }) => {
    const canvas = within(canvasElement);

    await userEvent.click(canvas.getByText(3));

    await expect(canvas.getByText(/11/i)).toBeInTheDocument();
    await expect(canvas.getByText(/15/i)).toBeInTheDocument();
  },
};
```

Storybook

Find components /

GUIDES

Angular

SHARED

Components

Button

Pagination

Documentation

Default

With Few Items

With Few Items

USERS

Users Page

« 1 2 3 4 5 »

Controls 10 Actions Visual Tests Interactions 3 Addons ×

PASS Scroll to end ↶ ↷ ⏪ ⏫ ⏭ ⏮ pagination.stories.ts

✓ userEvent.click(within(<div#storybook-root>).getByText(3))

✓ expect(within(<div#storybook-root>).getByText(/11/i)).toBeInTheDocument()

✓ expect(within(<div#storybook-root>).getByText(/15/i)).toBeInTheDocument()

**Exercise**

# Add-ons: Accessibility and Interactions

- Add the add-ons for accessibility and interactions
- In the Pagination stories add the following test to the default story
  - Test “100” is in the document
  - Test “results” is in the document
- Add the following tests for the “With Few Items” story
  - Click on the button with “3” and verify the page changed by testing that
    - “11” and “15” are in the document



Develop

Document

Test