StoryBook with Angular

bit.ly/w-sb-angular

bit.ly/w-sb-angular

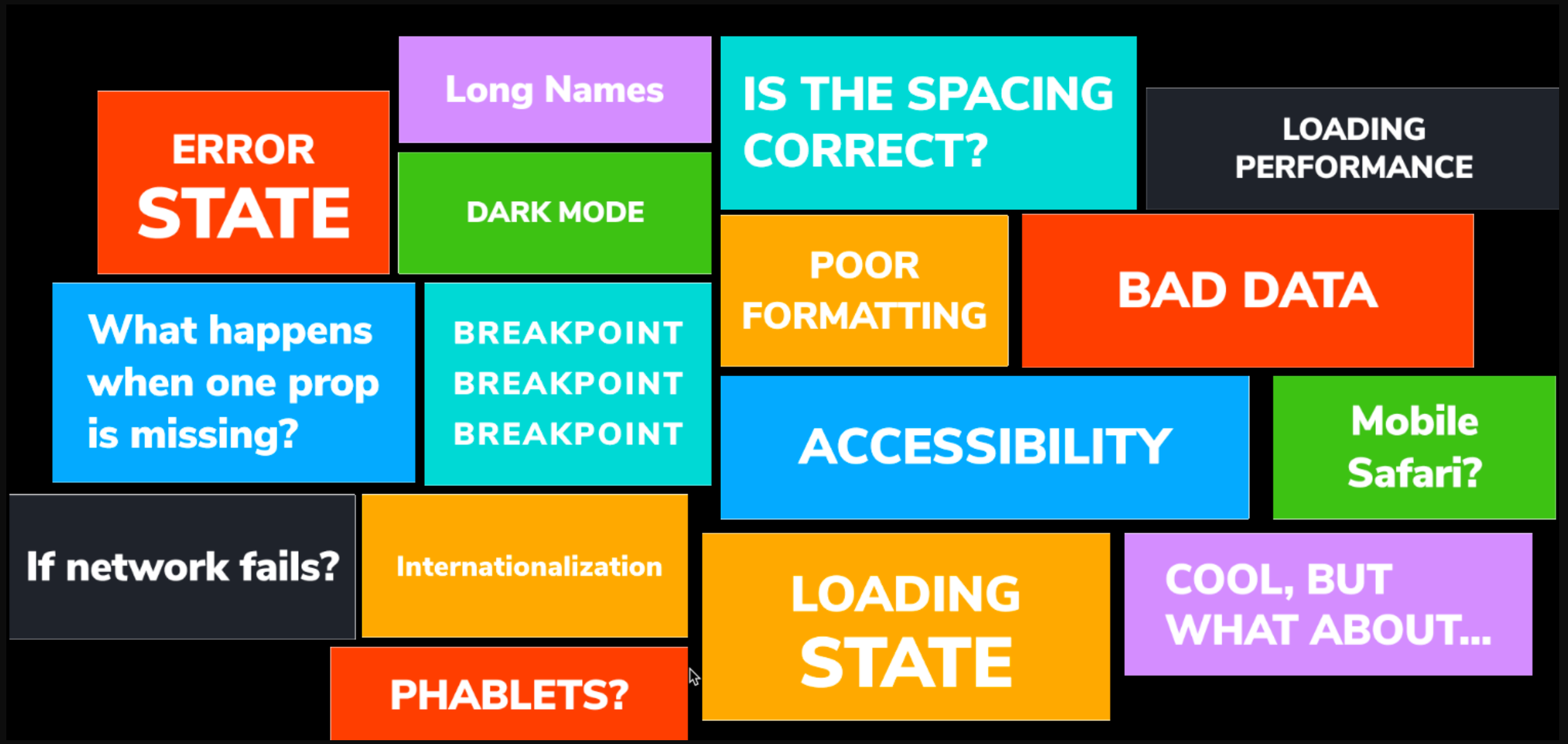


*Image from a presentation by StoryBook

Develop

Document

Test

Exercises

Story for a presentational component

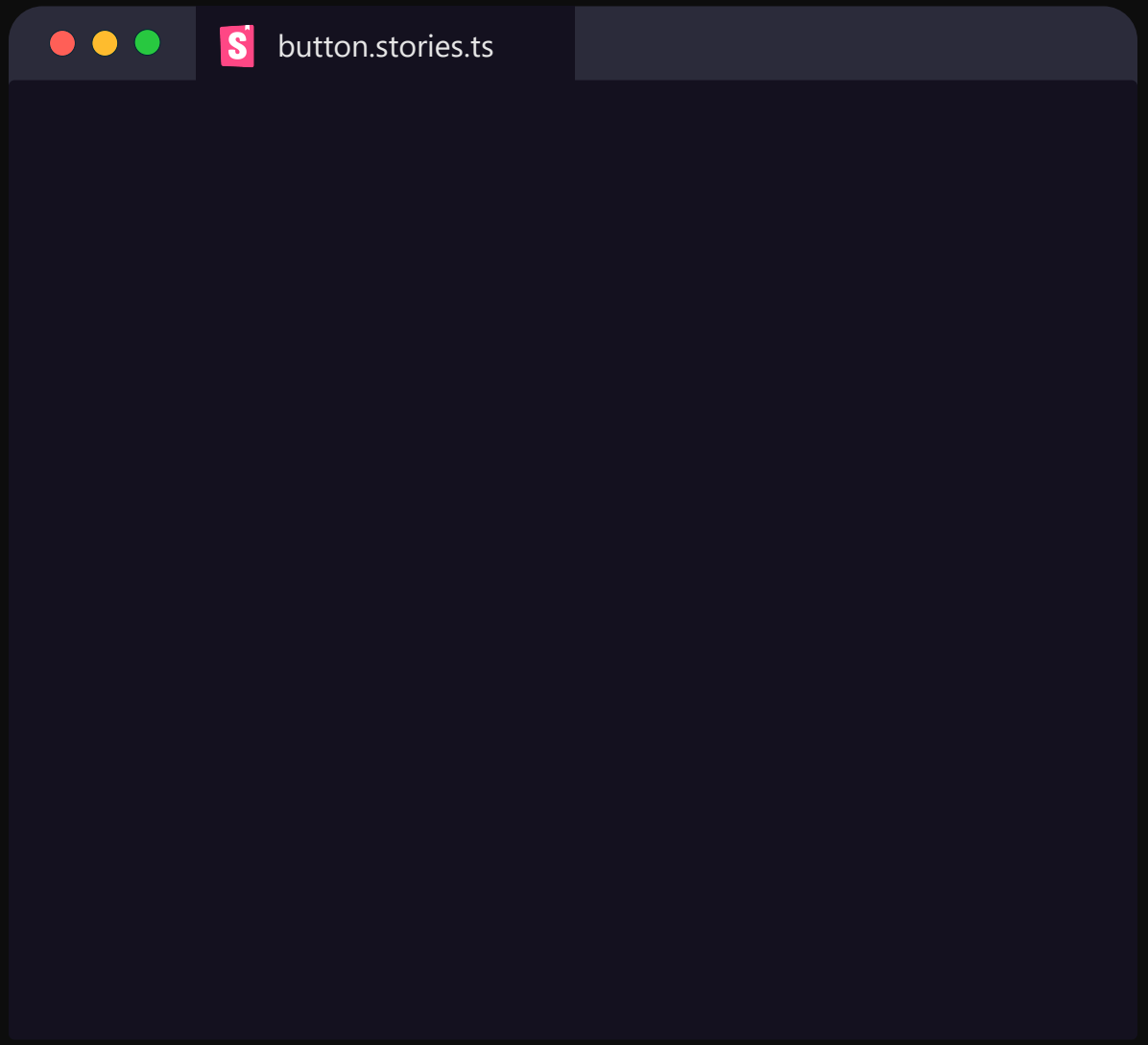
Story for a container component

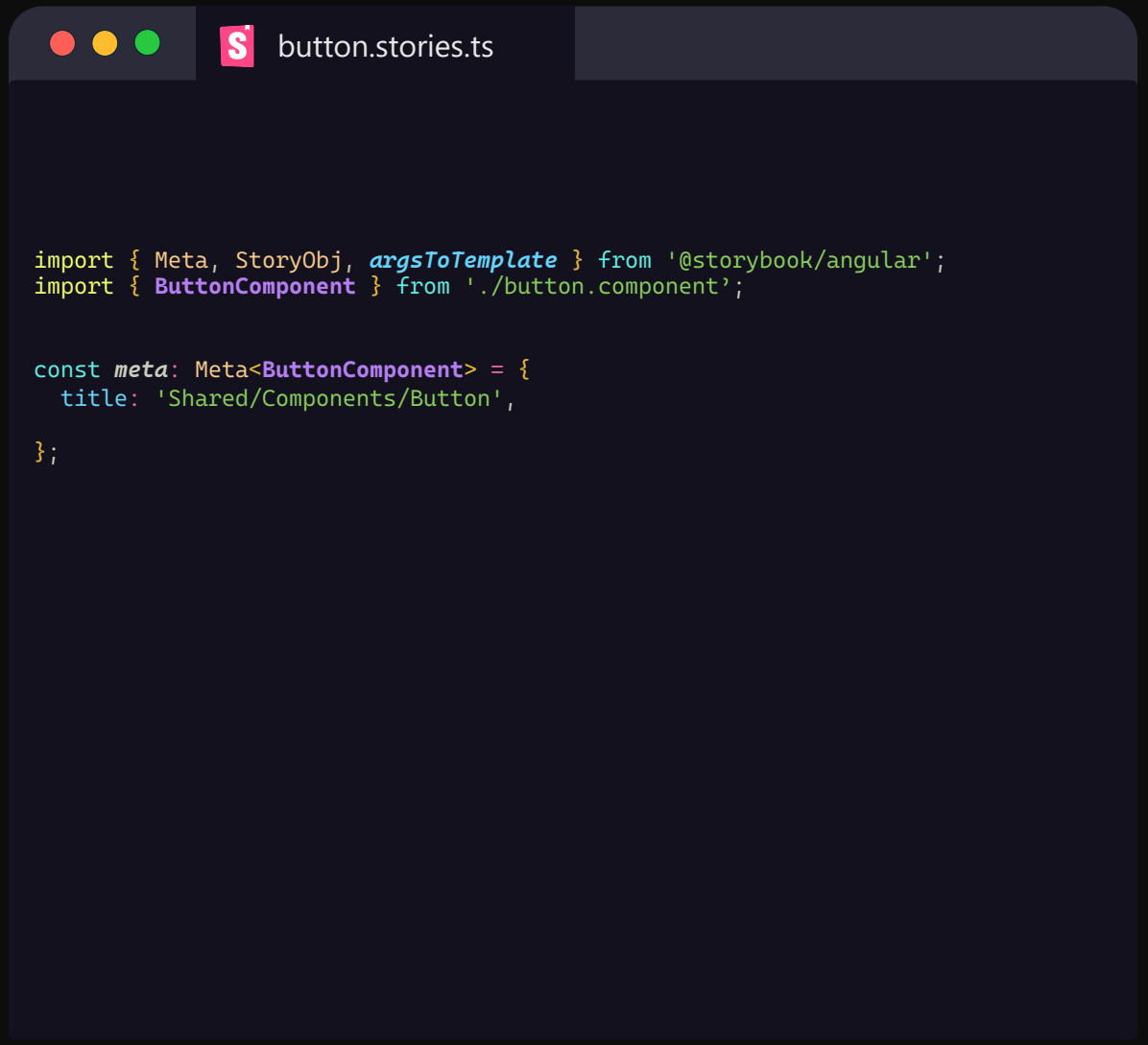
Creating a guideline

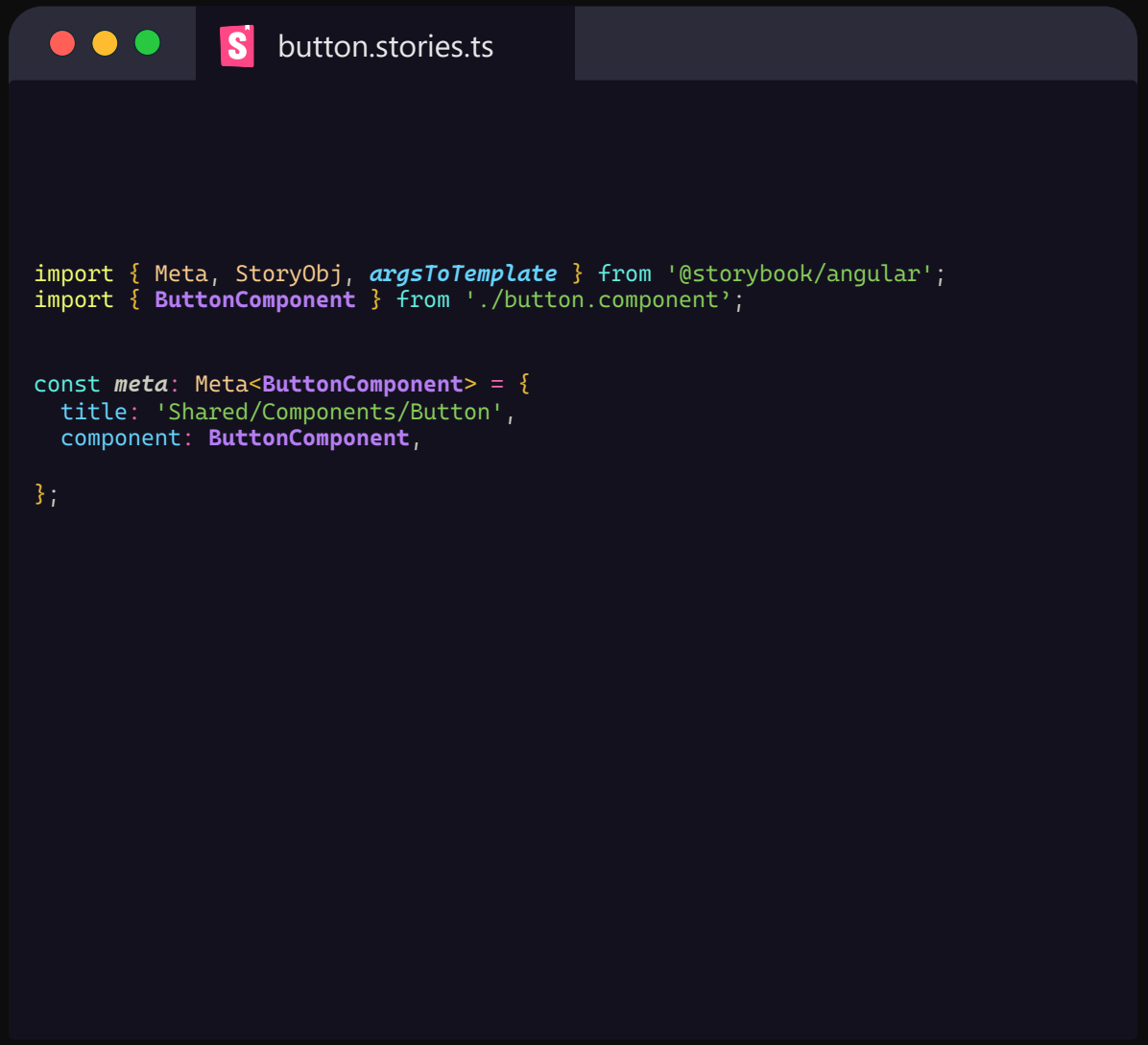
Adding custom documentation for a component

Extra - Tests and Accessibility addons


Button!


button.stories.ts


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

```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>`,
  }),
};
```

button.stories.ts

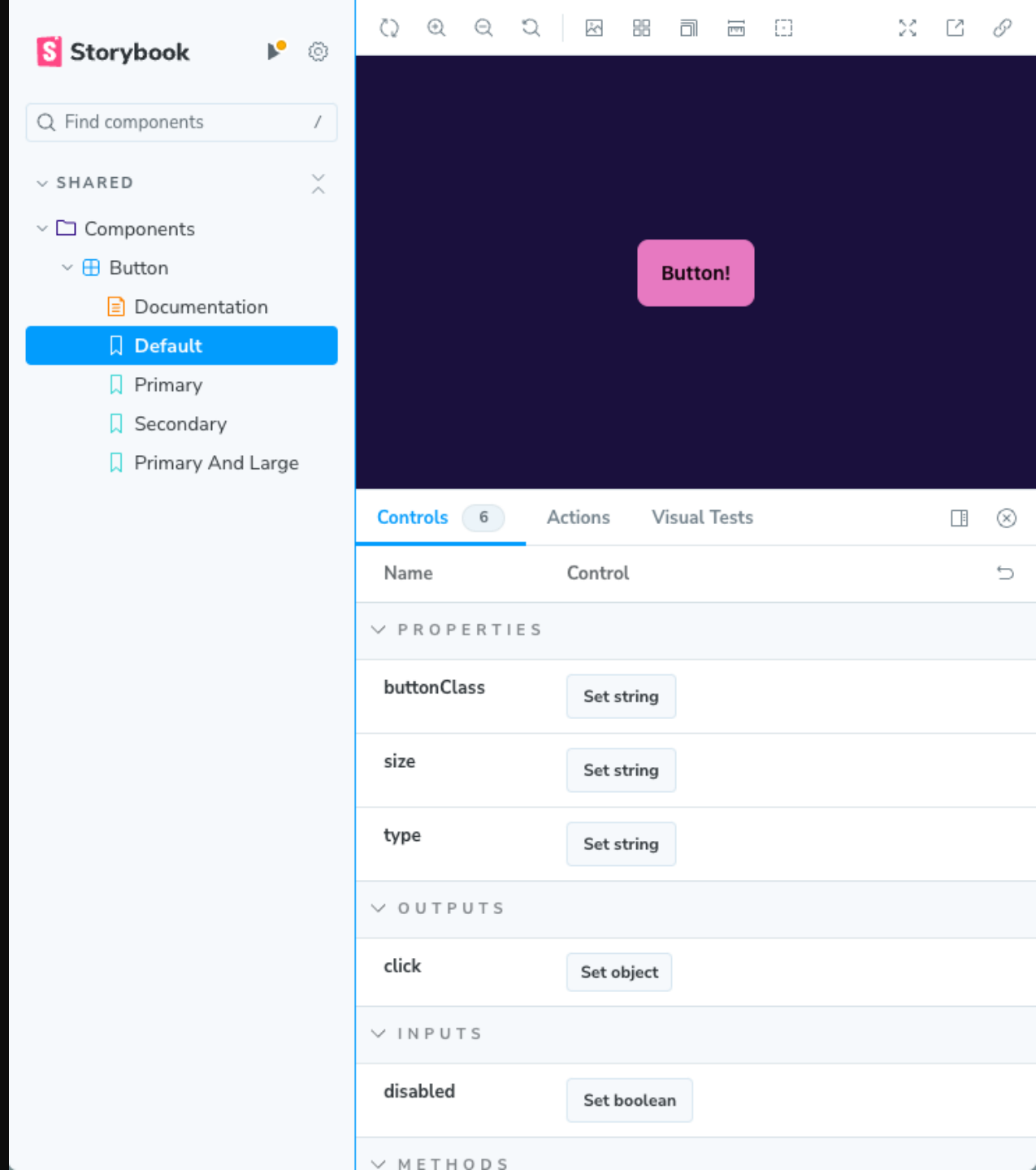
```ts
import { Meta, StoryObj, argsToTemplate } from '@storybook/angular';
import { ButtonComponent } from './button.component';


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>;
```

button.stories.ts

```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 = {};
```

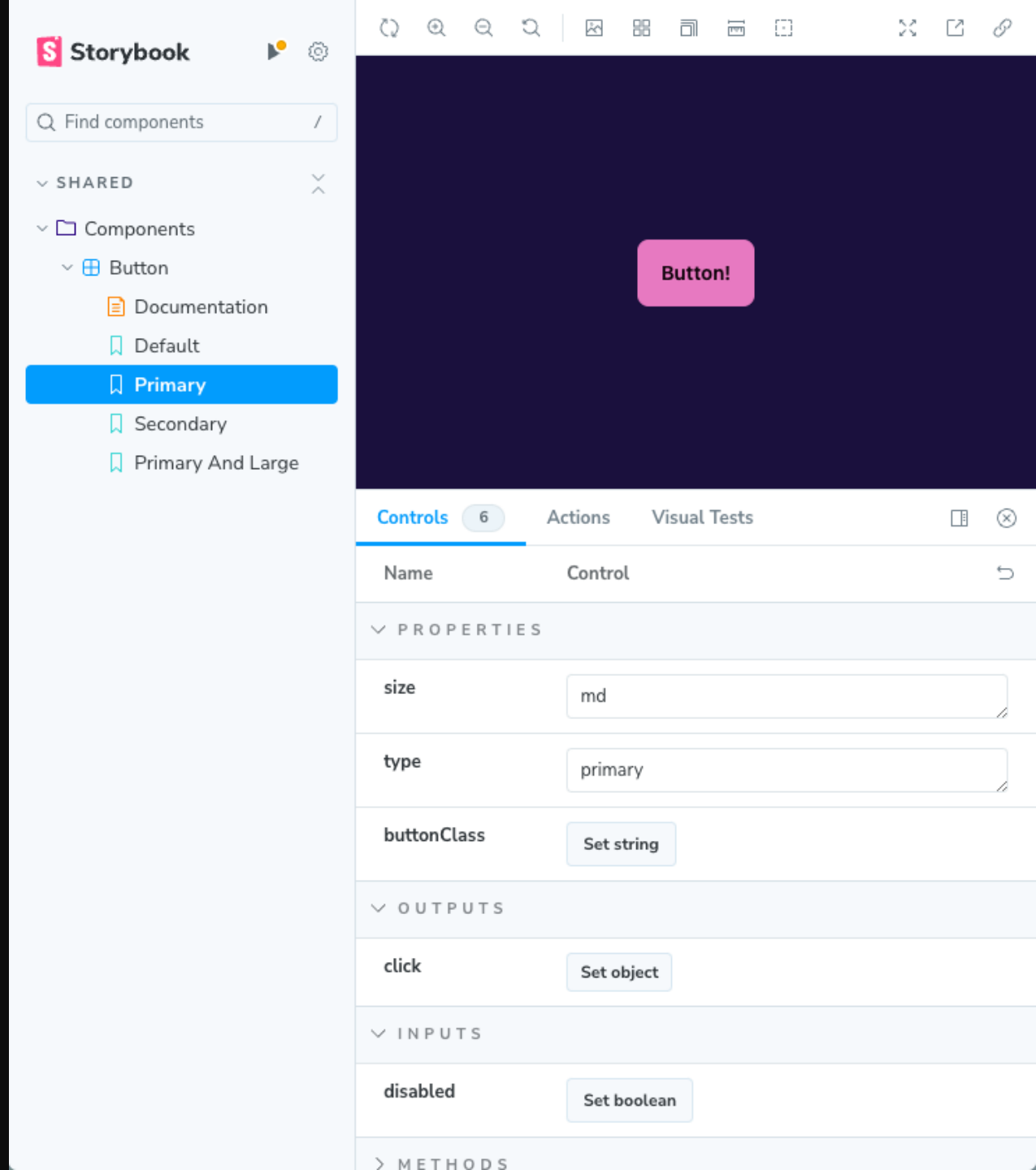
button.stories.ts

```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 = {};
```



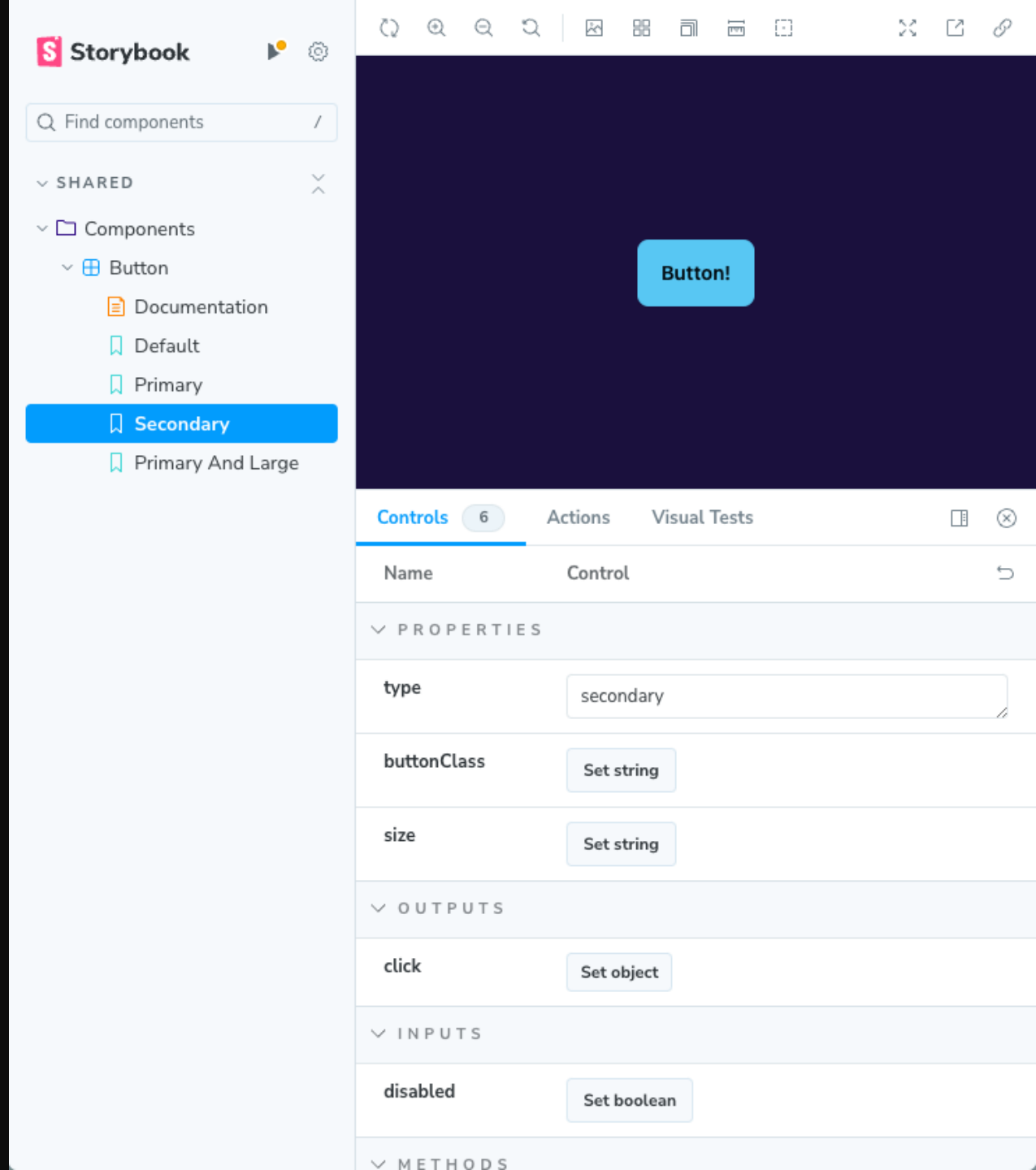
button.stories.ts

```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',
  },
};
```



button.stories.ts

```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',
  },
};
```



button.stories.ts

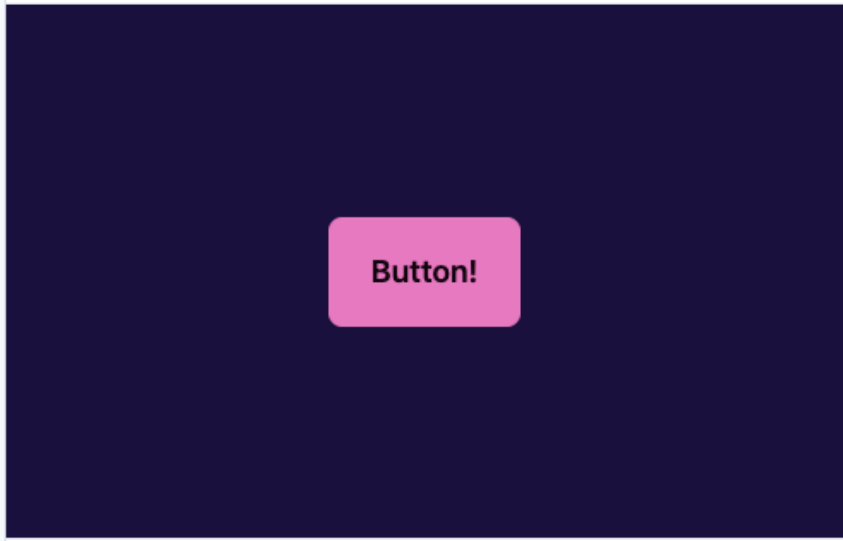
```ts
import { Meta, StoryObj, argsToTemplate } from '@storybook/angular';
import { ButtonComponent } from './button.component';


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

Storybook

Find components /

SHARED

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



Controls 6 | Actions | Visual Tests

Name	Control
PROPERTIES	
size	lg
type	primary
buttonClass	Set string
OUTPUTS	
click	Set object
INPUTS	
disabled	Set boolean
METHODS	

Exercise Story for a presentational component

- Add Story for Pagination component with the following options

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

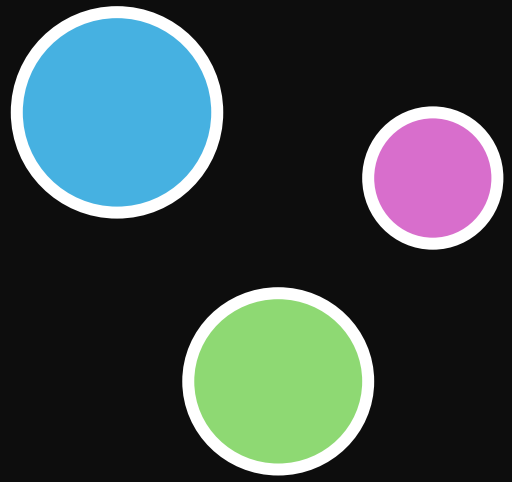
- Set the location of the story to be on `Shared/Components`

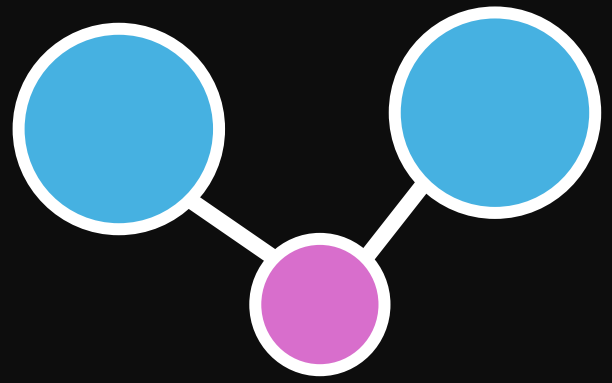
- Add a story of `WithFewItems` that has only 10 total items

7:00

1-story-for-presentational-component

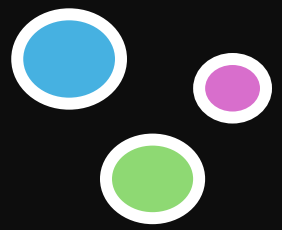
Develop

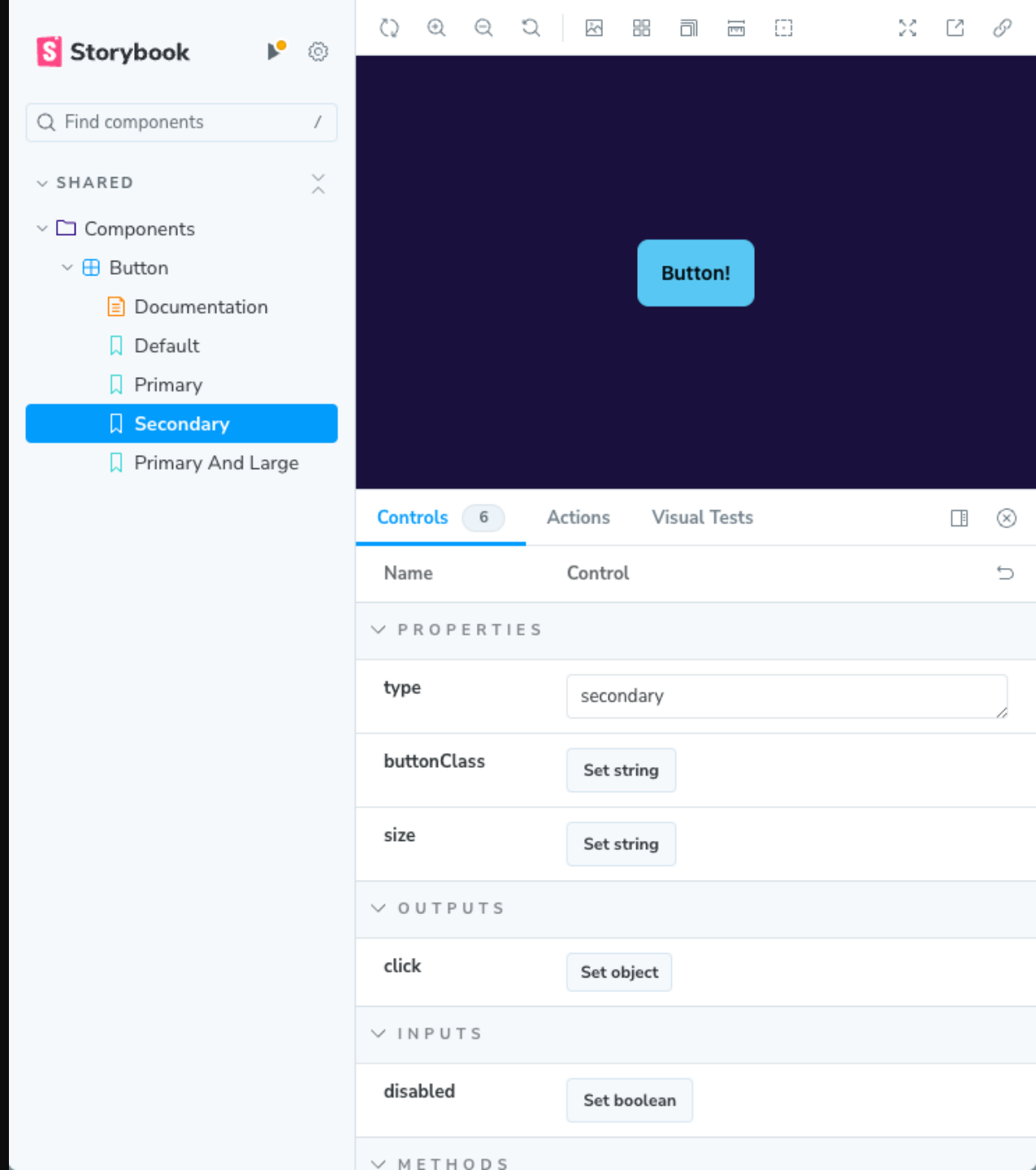




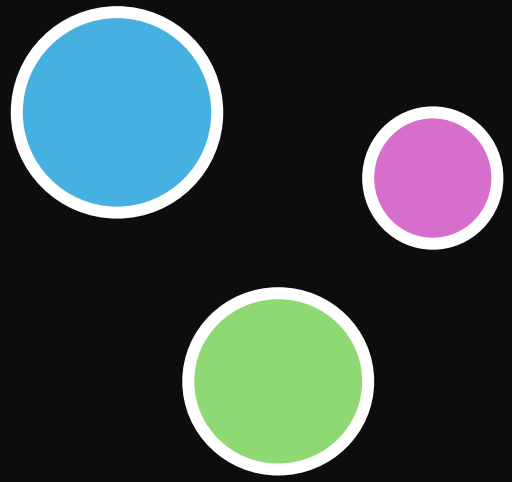
Develop

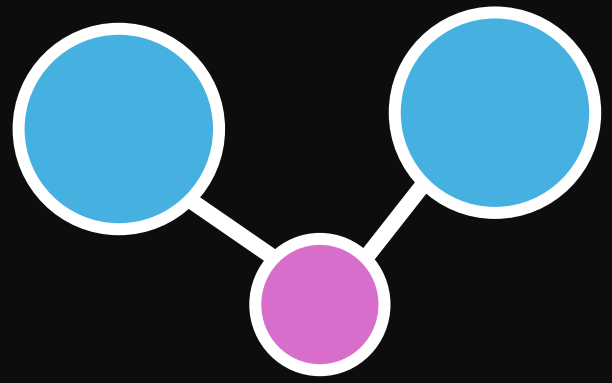
Atoms





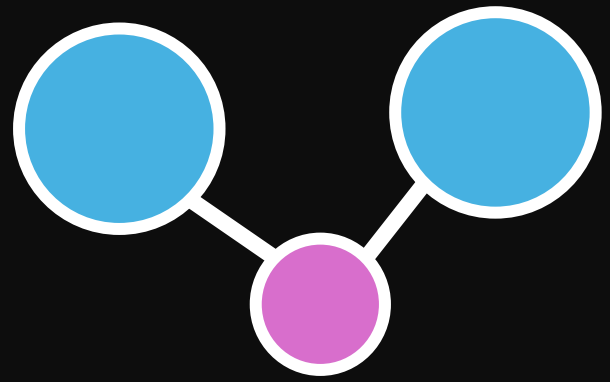
Develop

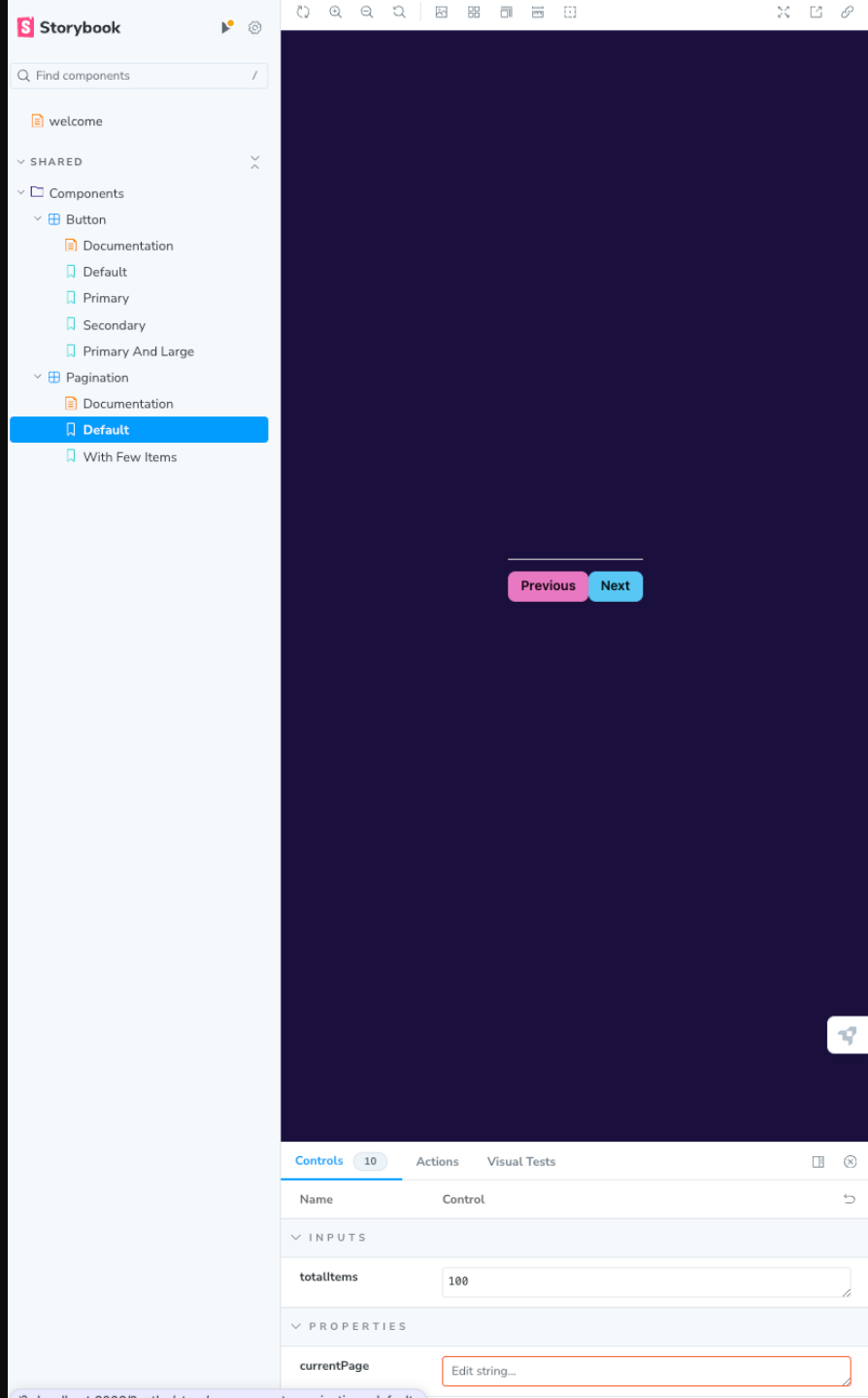




Develop

Molecules





Try States

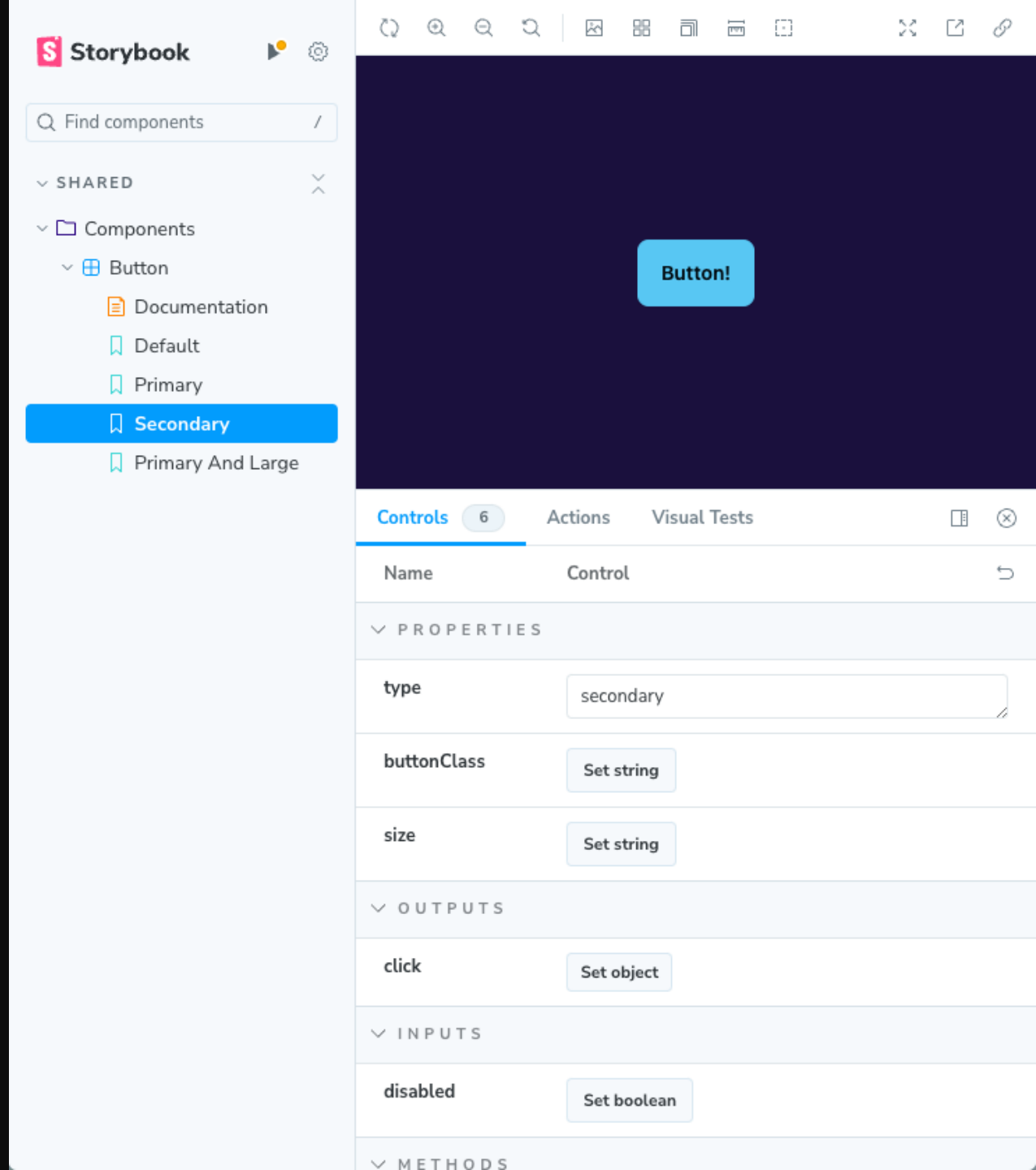
Try States

Button

- Primary
- Secondary
- Primary and Large

Pagination

- With 100 items
- With Few Items



Smart Component

Smart Component

Mocks File

API Service Mock

Mocks File

user-page.mocks.ts

```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,
};
```

Mocks File

user-page.mocks.ts

```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

user-page.mocks.ts

```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

user-page.mocks.ts

```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

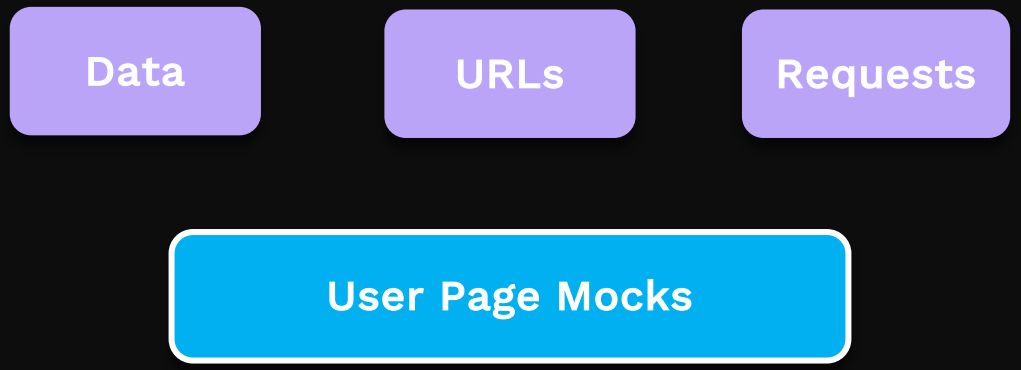
URLs

Requests

Mocks File

user-page.mocks.ts

```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,
};
```



API Service Mock

user-page.stories.ts

```ts
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

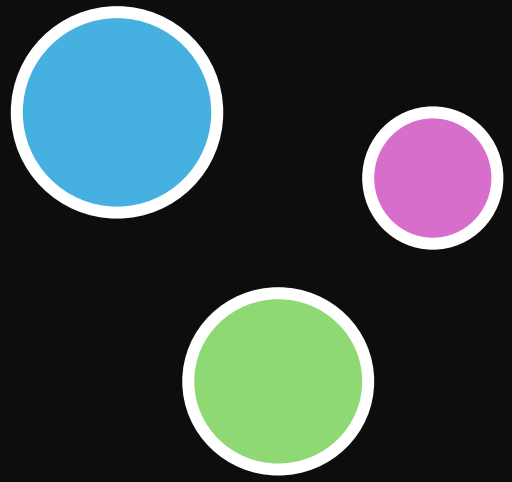
- Set the location of the story to be on `Users/Users Page`

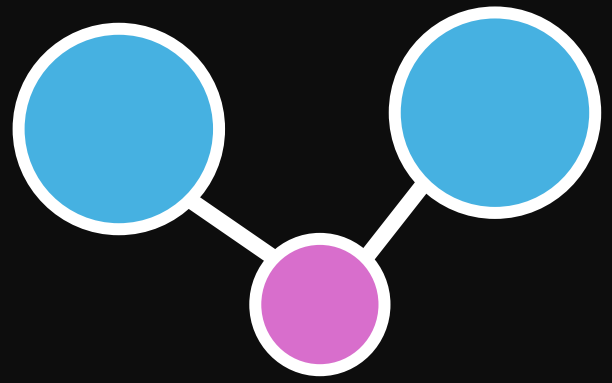
- Add a story of that uses the user page mocks with `setApiMocks`

5:00

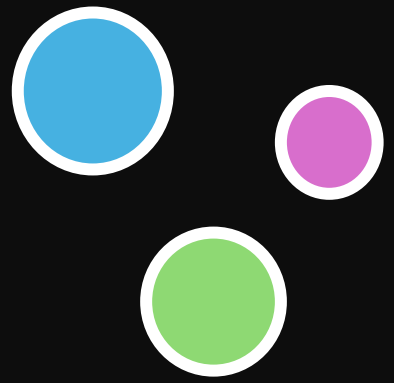
2-story-for-smart-component

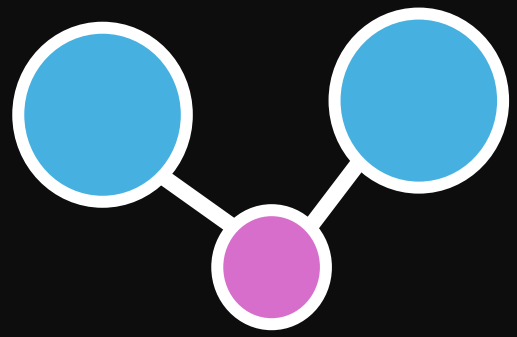
Develop

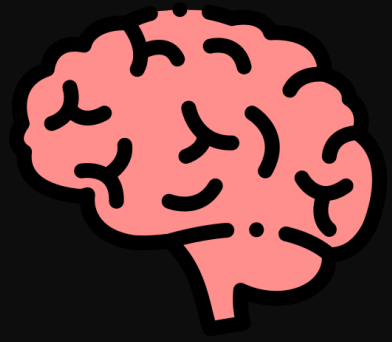




Develop







Smart Component

Smart Component

Mocks File

API Service Mock

Smart Component

Mocks File

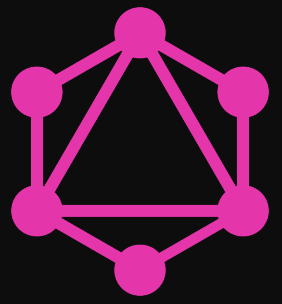
~~API Service Mock~~

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

Modal

Overlay

Section

Tab

Container

Develop

Document

Test

StoryBook Configuration

main.ts

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

const config: StorybookConfig = {
  stories: [
    '../src/**/*.mdx',
    '../src/**/*.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;
```

main.ts

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

const config: StorybookConfig = {
  stories: [
    '../src/**/*.mdx',
    '../src/**/*.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;
```

angular-guidelines.mdx

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

<Meta title="Guides/Angular" />

# Angular Guidelines

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

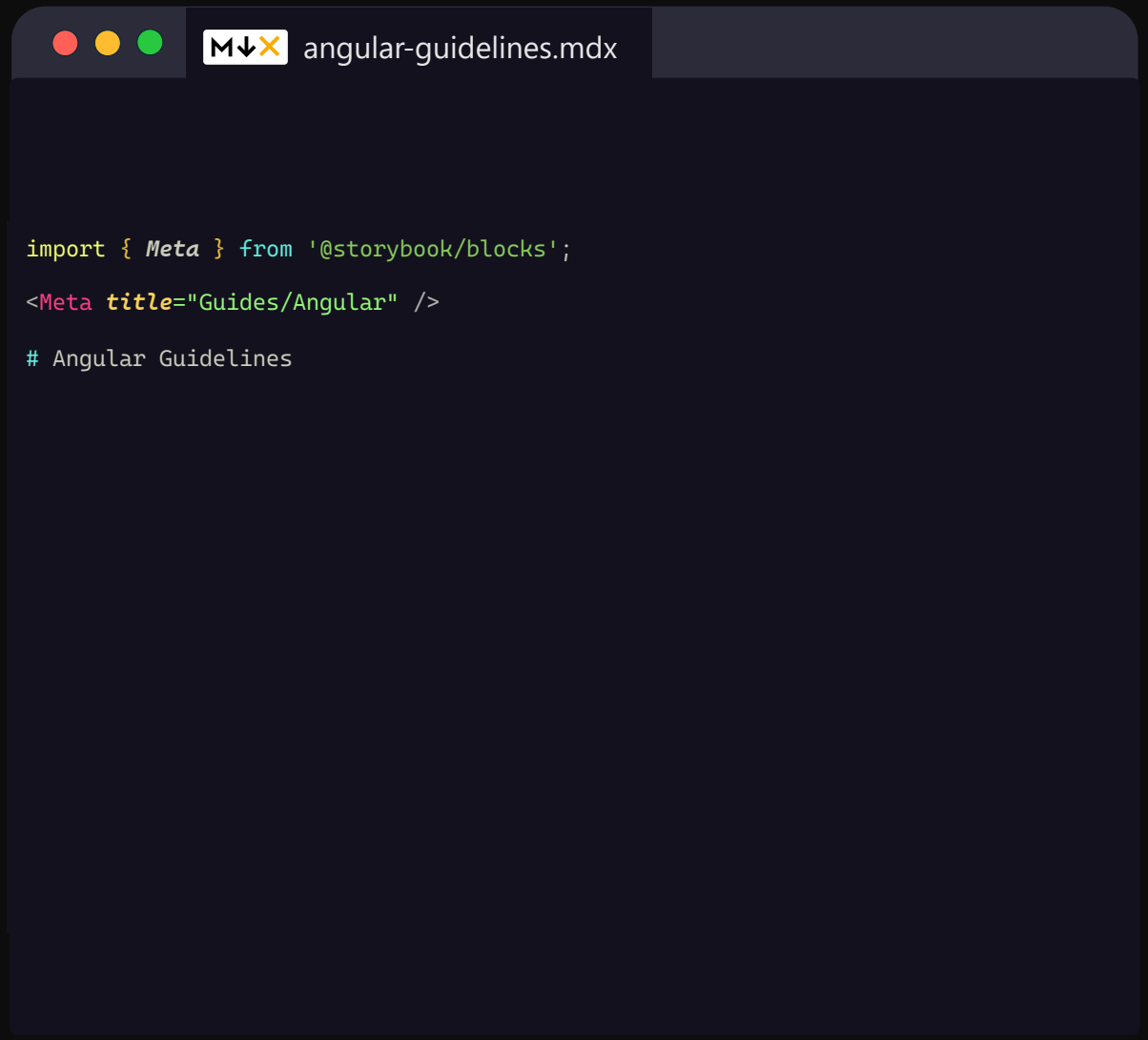
## Components

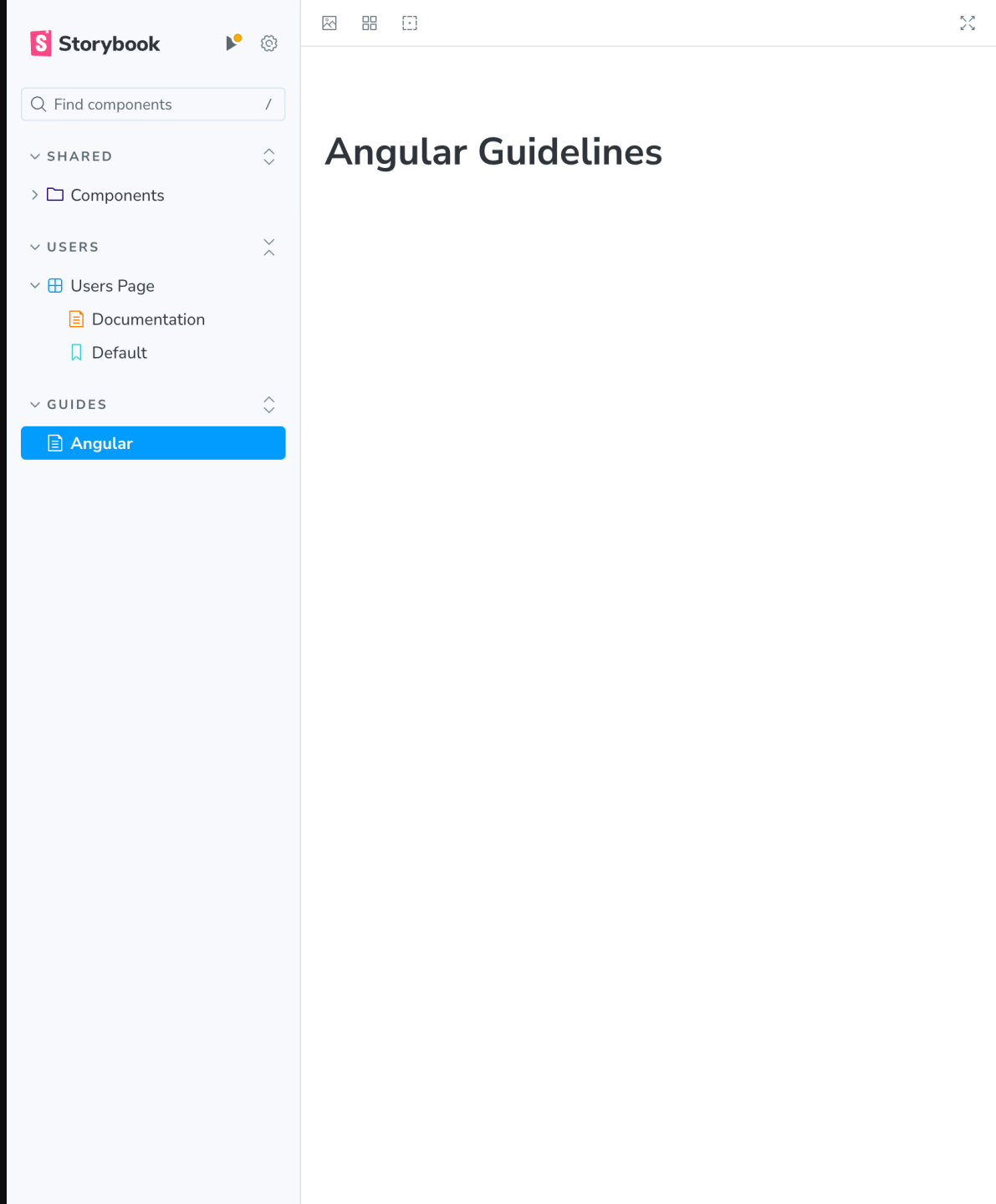
### Do not use services with state in presentational components

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)
}
```
```


angular-guidelines.mdx
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

angular-guidelines.mdx

```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)
}
```
```

Storybook

Find components /

SHARED

Components

USERS

Users Page

Documentation

Default

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

```
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

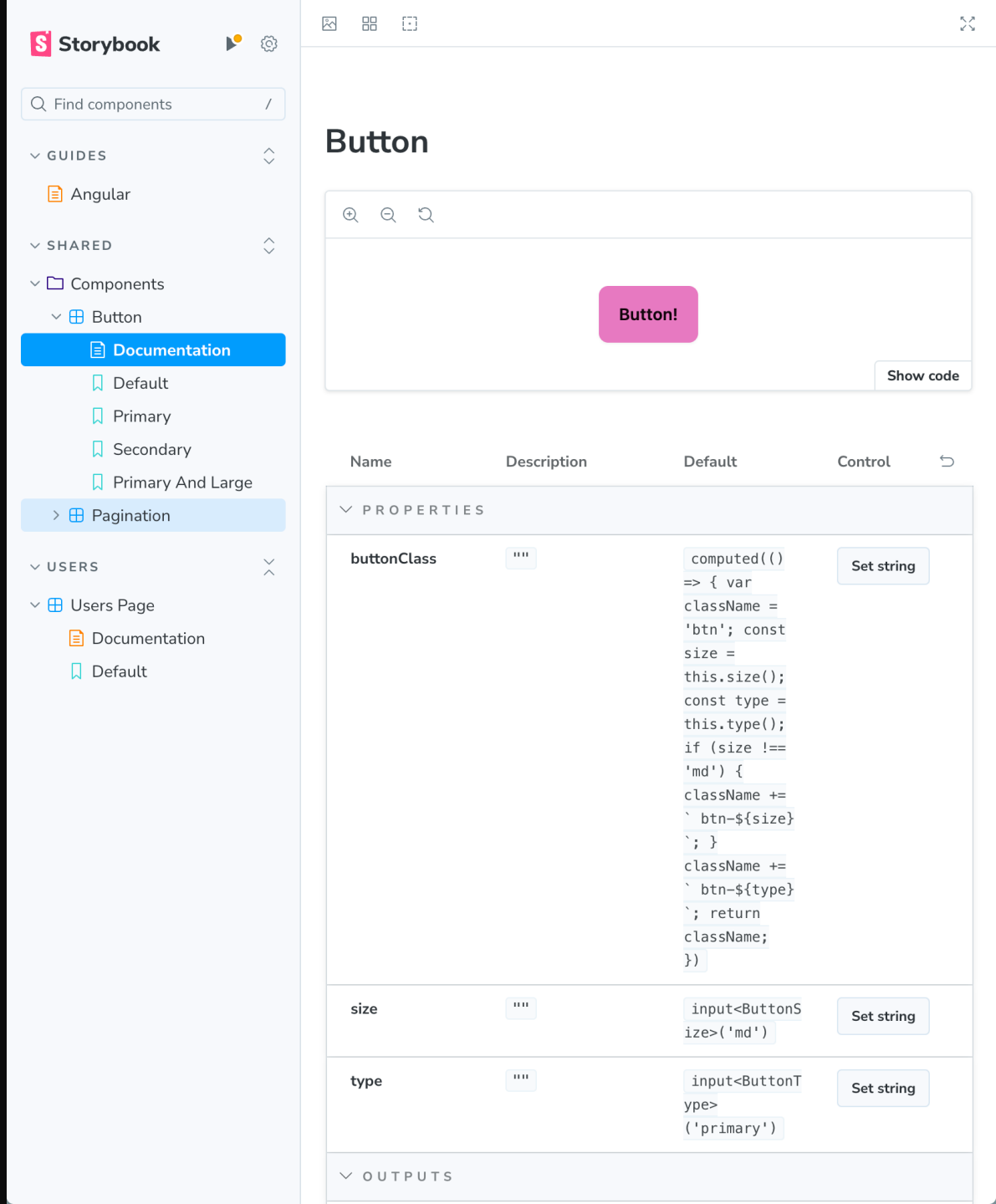
StoryBook Configuration

main.ts

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

const config: StorybookConfig = {
  stories: [
    '../src/**/*.mdx',
    '../src/**/*.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;
```

main.ts

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

const config: StorybookConfig = {
  stories: [
    '../src/**/*.mdx',
    '../src/**/*.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;
```



button.stories.ts

```ts
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',
  },
};
```

Storybook

Find components /

GUIDES

Angular

SHARED

Components

Button

Documentation

Default

Primary

Secondary

Primary And Large

Pagination

USERS

Users Page

Documentation

Default

Button

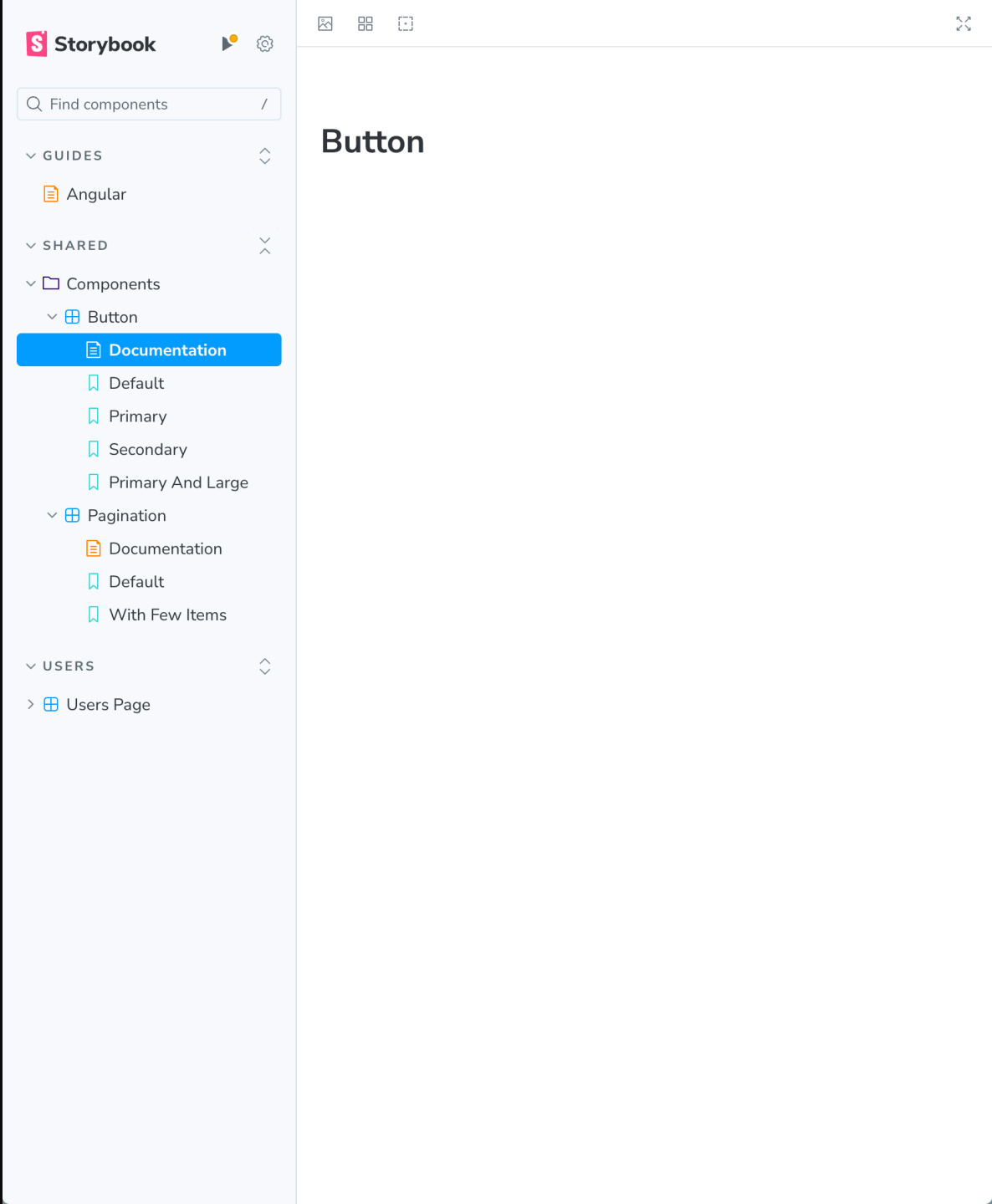
Button!

Show code

Name	Description	Default	Control
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<ButtonSize>('md')	Set string
type	""	input<ButtonType>('primary')	Set string
OUTPUTS			

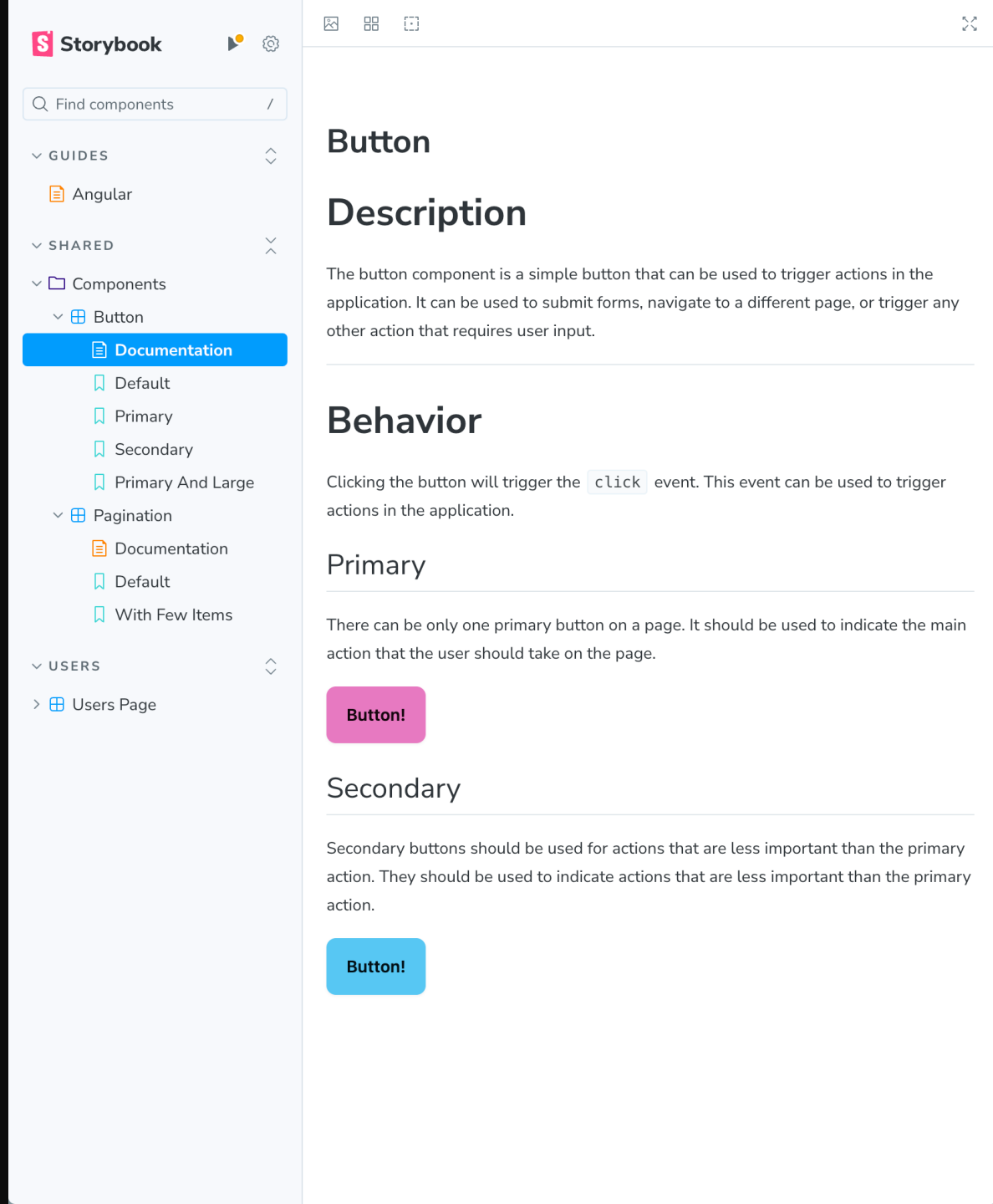
Button.mdx

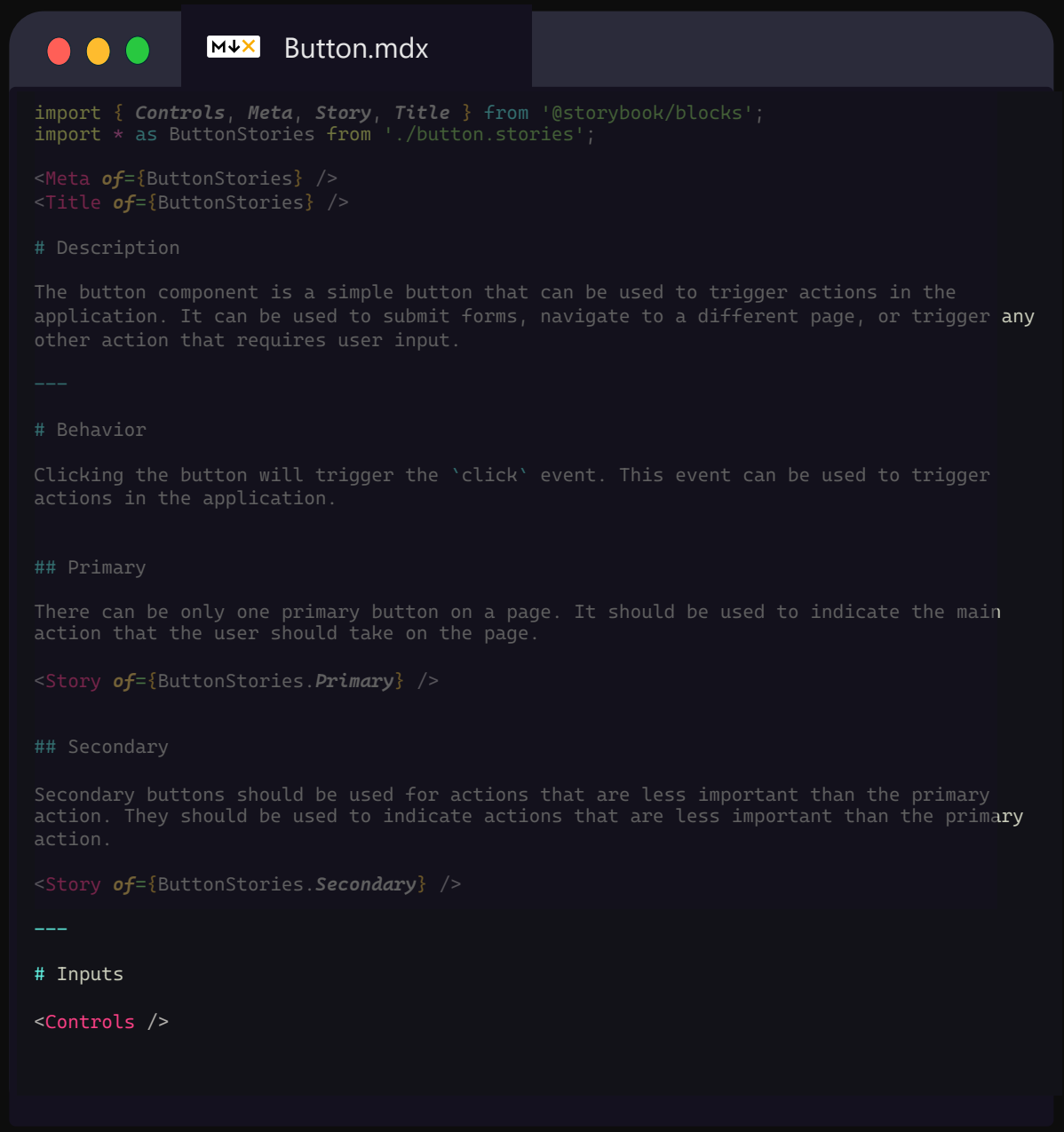
```mdx
import { Controls, Meta, Story, Title } from '@storybook/blocks';
import * as ButtonStories from './button.stories';

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



Button.mdx

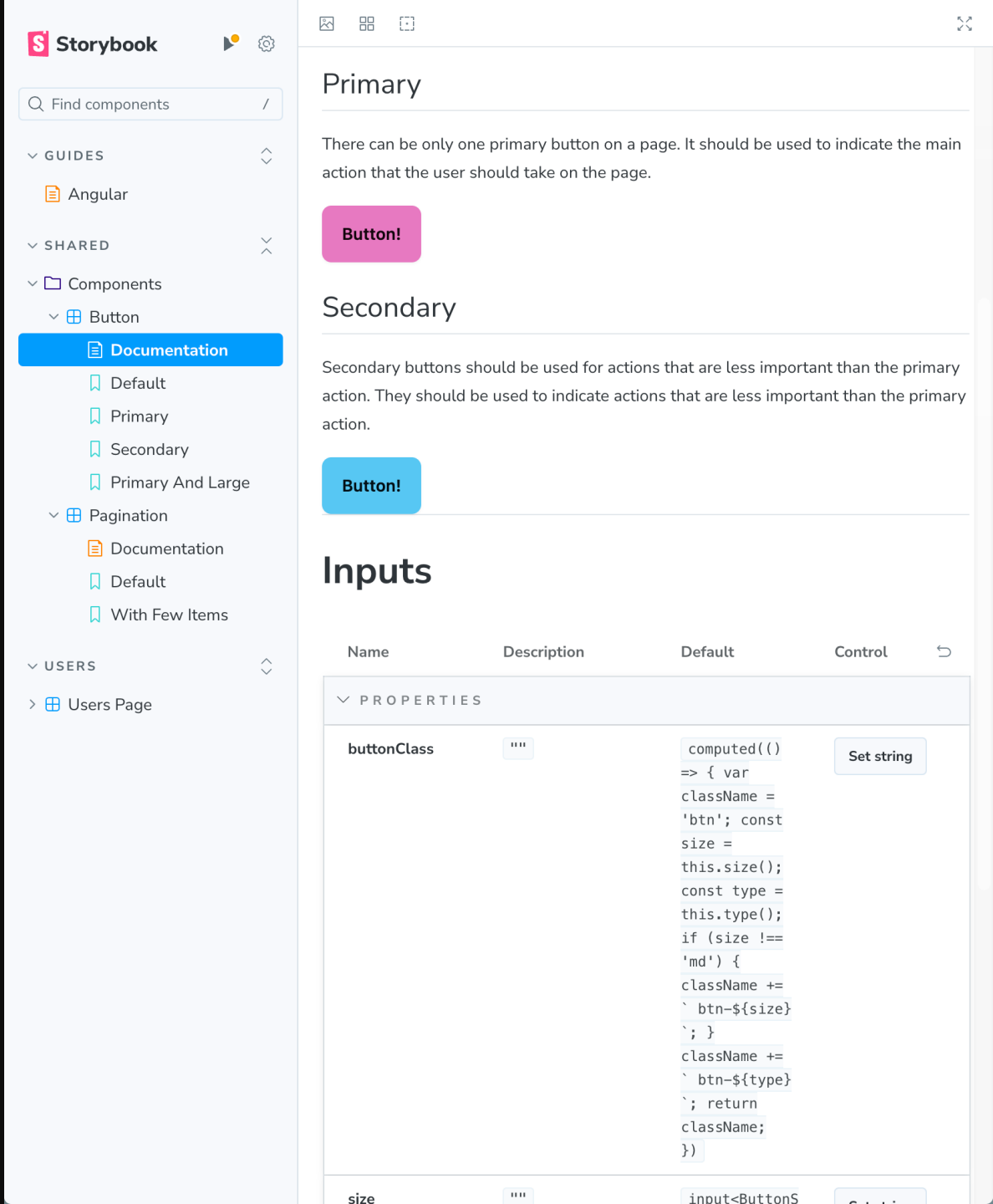
```mdx
import { Controls, Meta, Story, Title } from '@storybook/blocks';
import * as ButtonStories from './button.stories';

<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} />
```




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

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

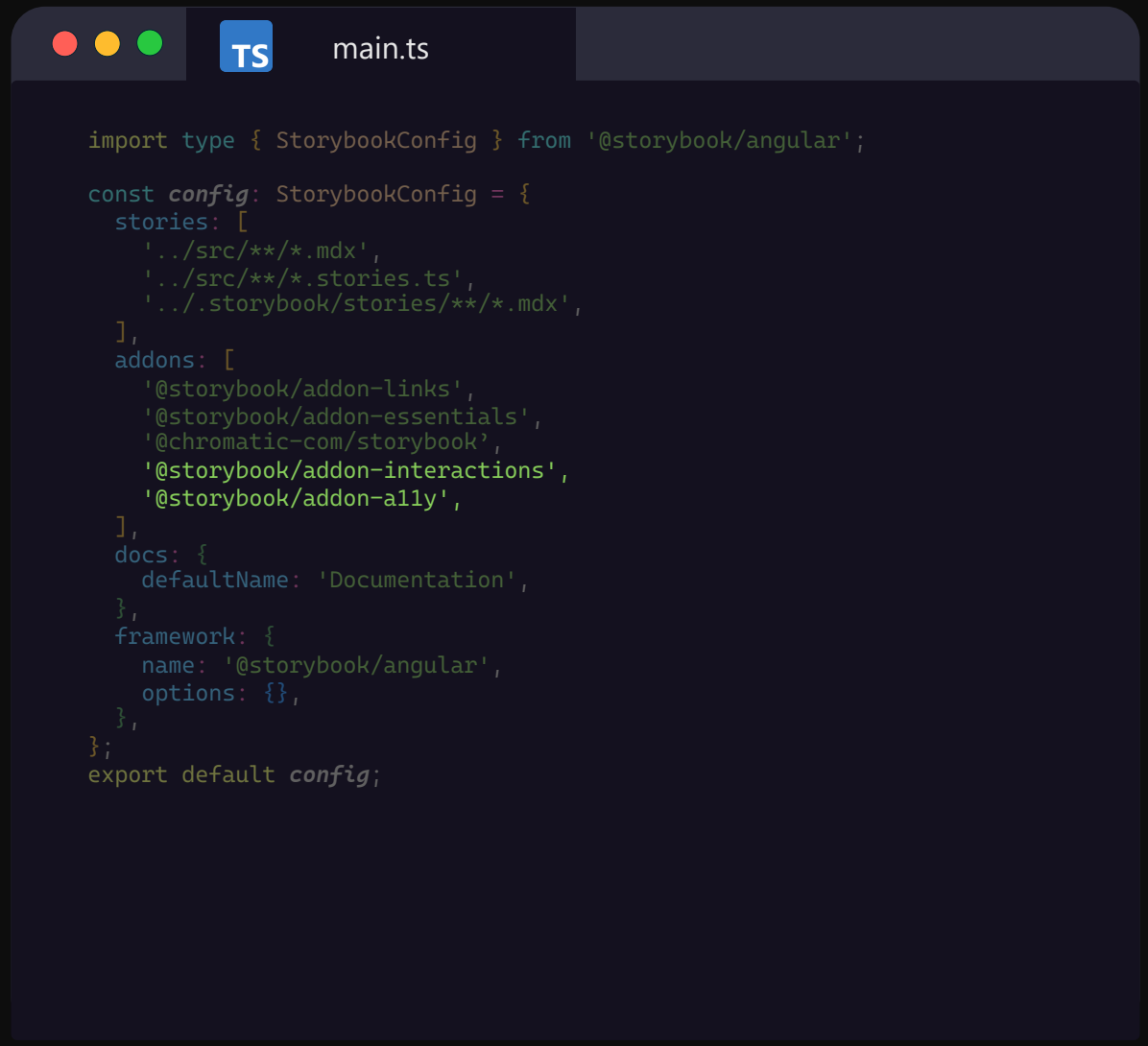
Accessibility

Interactions

Addons

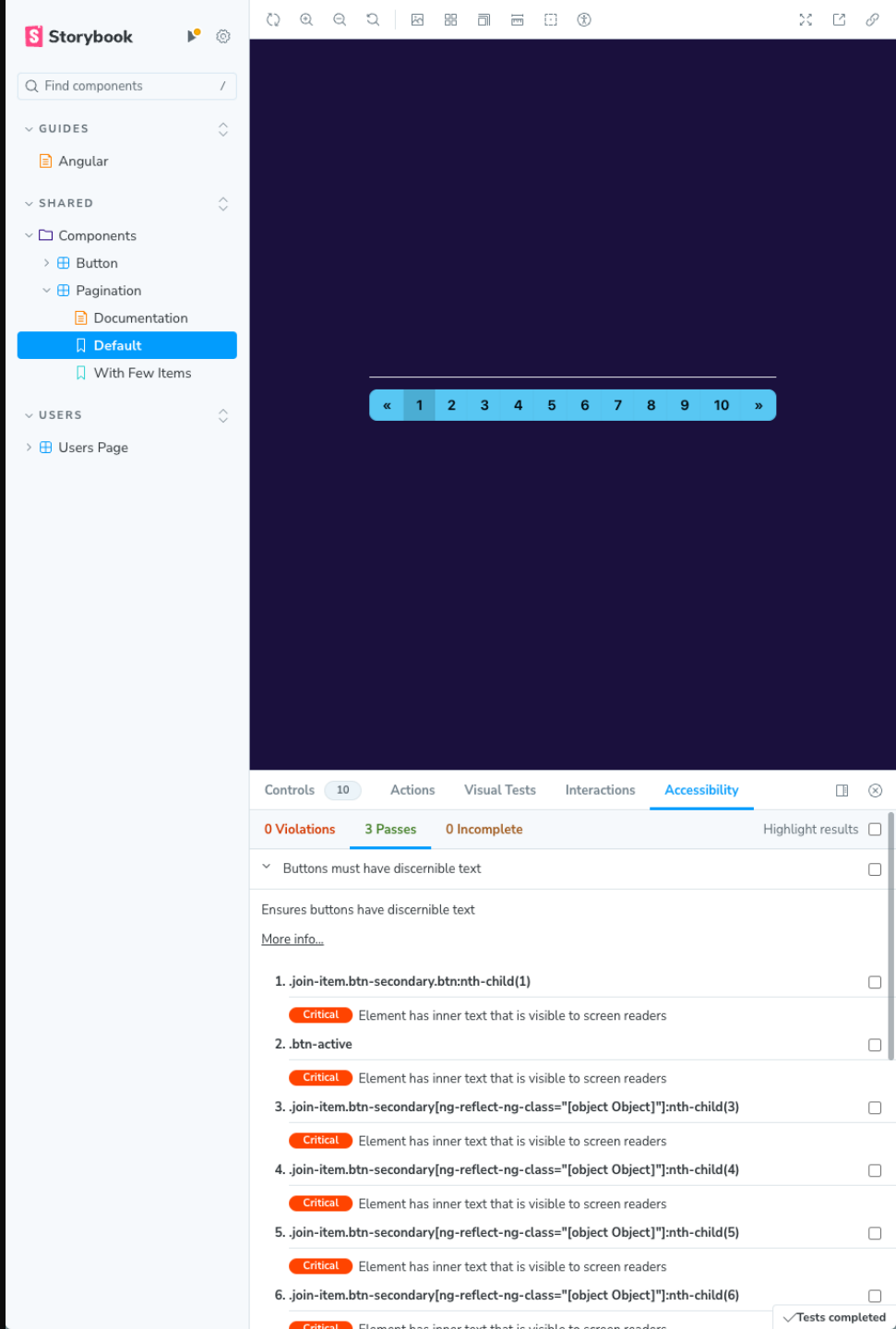
main.ts

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

const config: StorybookConfig = {
  stories: [
    '../src/**/*.mdx',
    '../src/**/*.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;
```


main.ts
import type { StorybookConfig } from '@storybook/angular';
const config: StorybookConfig = {
stories: [
'../src/**/*.mdx',
'../src/**/*.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;

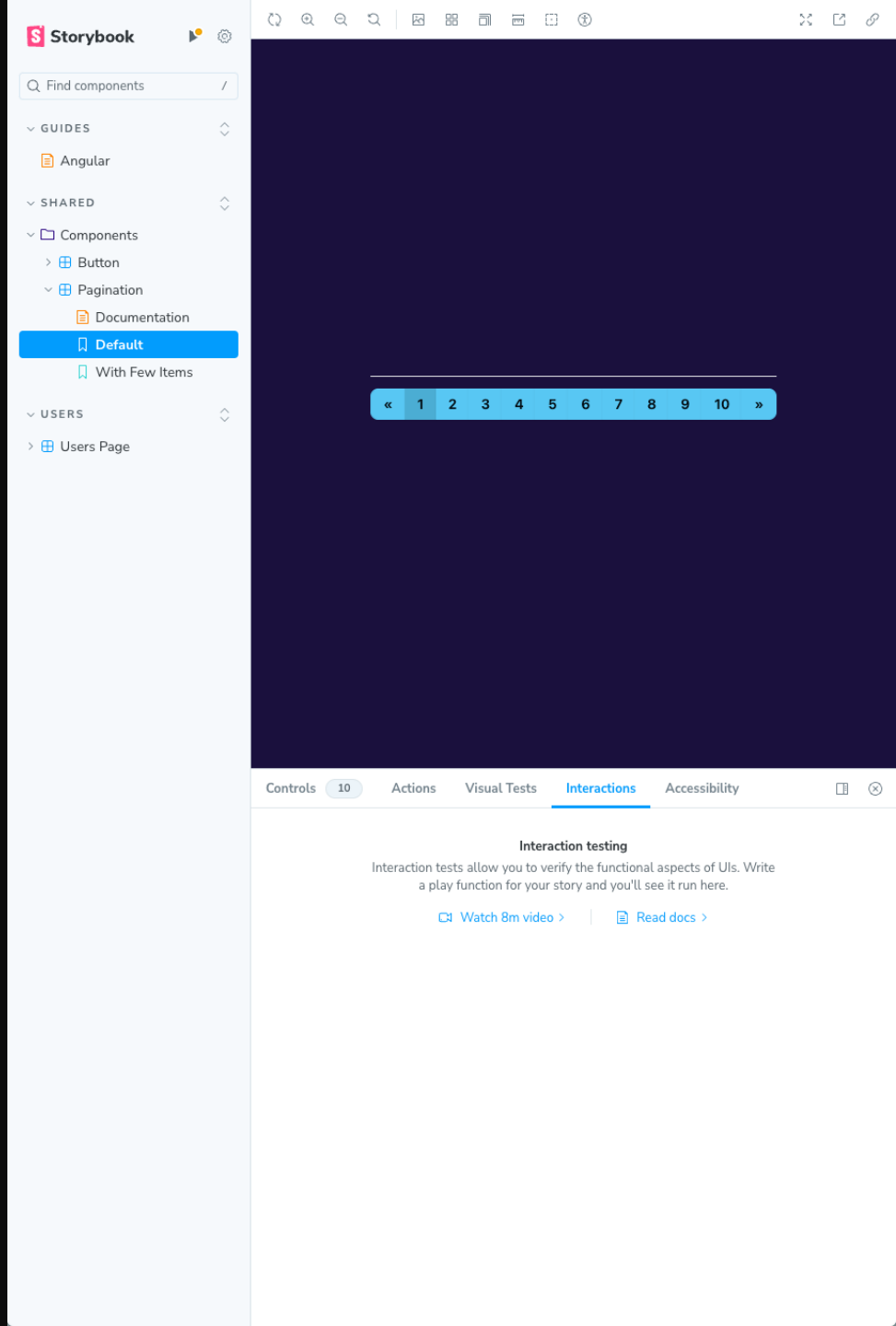
main.ts

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

const config: StorybookConfig = {
  stories: [
    '../src/**/*.mdx',
    '../src/**/*.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;
```



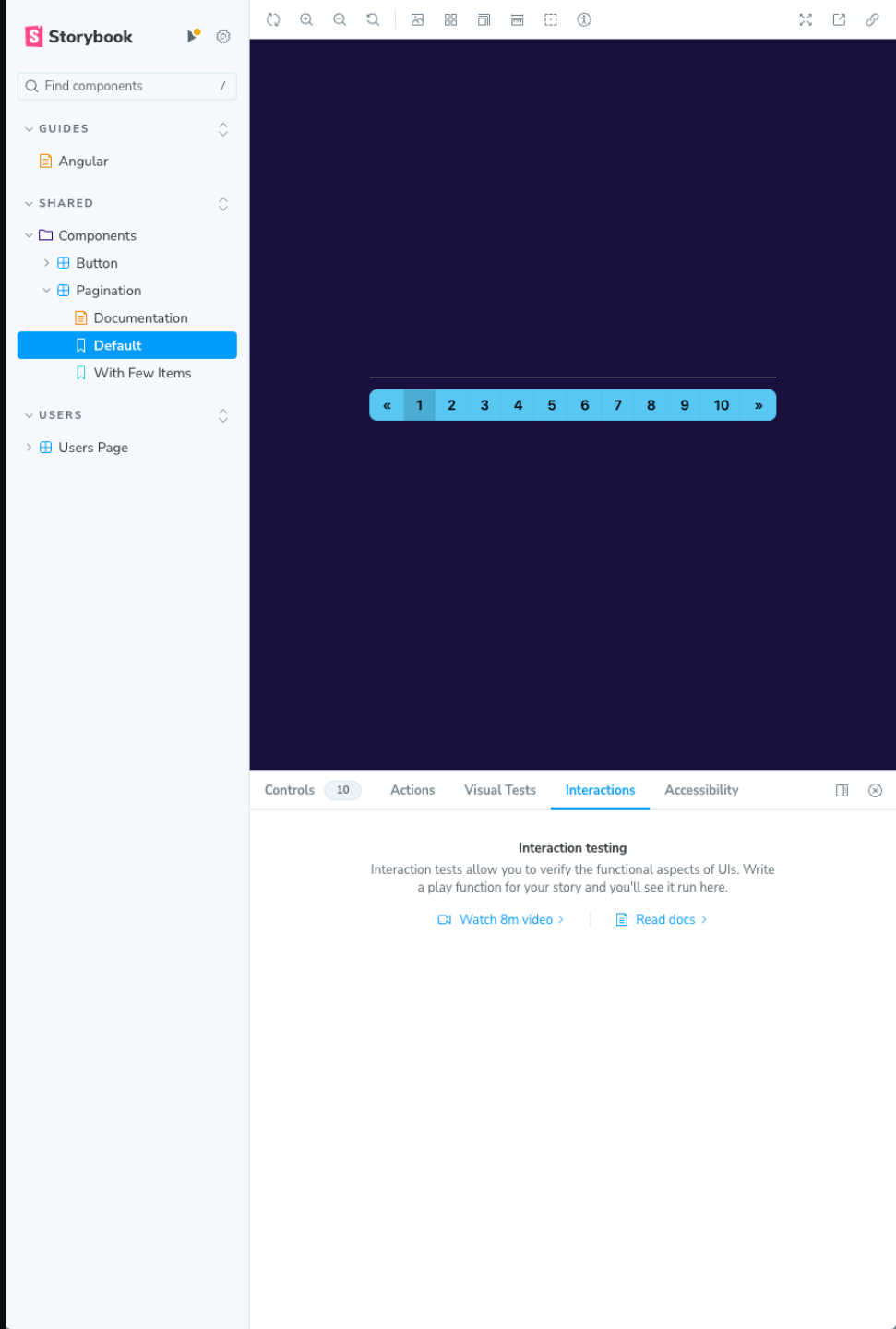
main.ts

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

const config: StorybookConfig = {
  stories: [
    '../src/**/*.mdx',
    '../src/**/*.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;
```



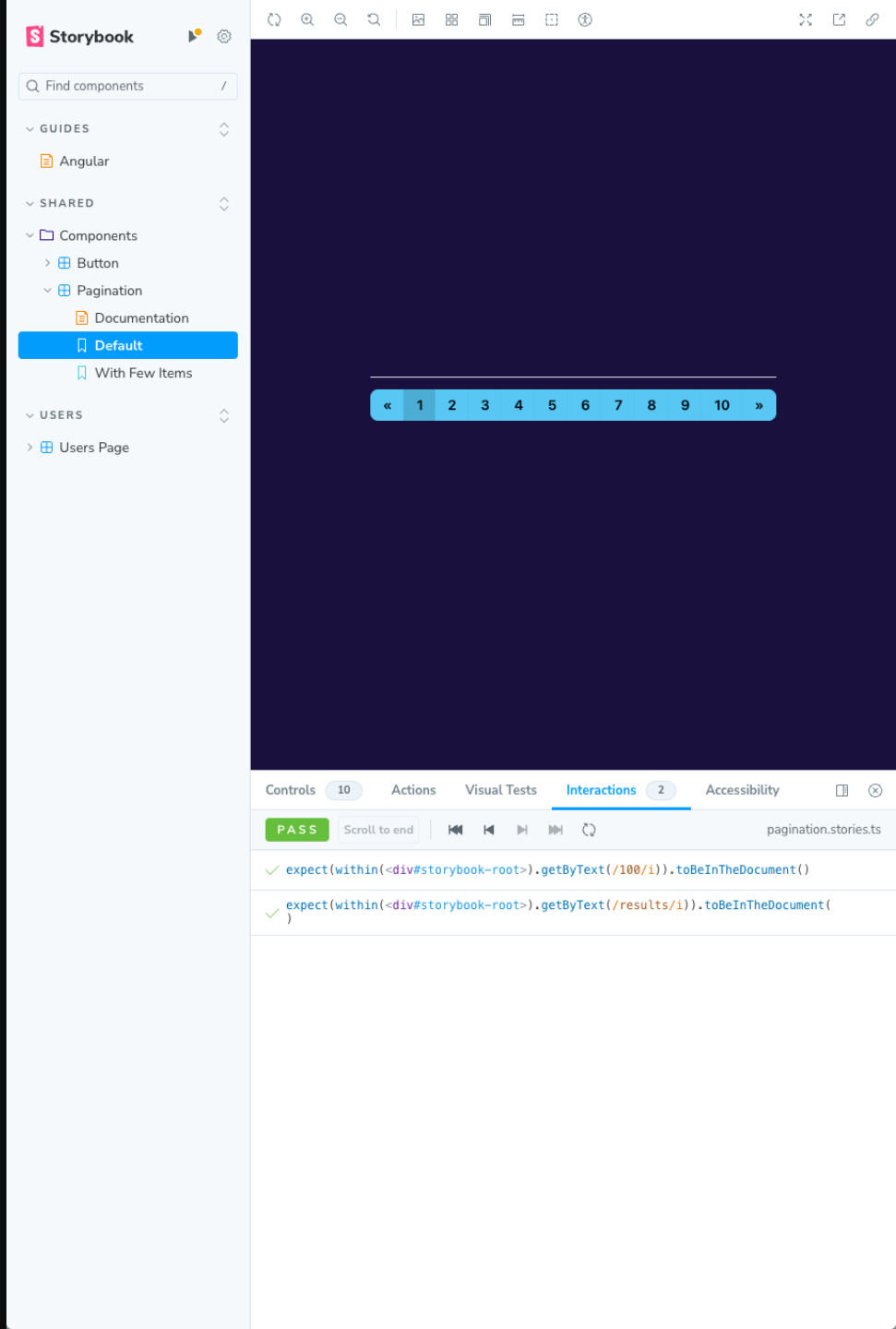
Pagination.stories.ts

```ts
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,
  },
};
```



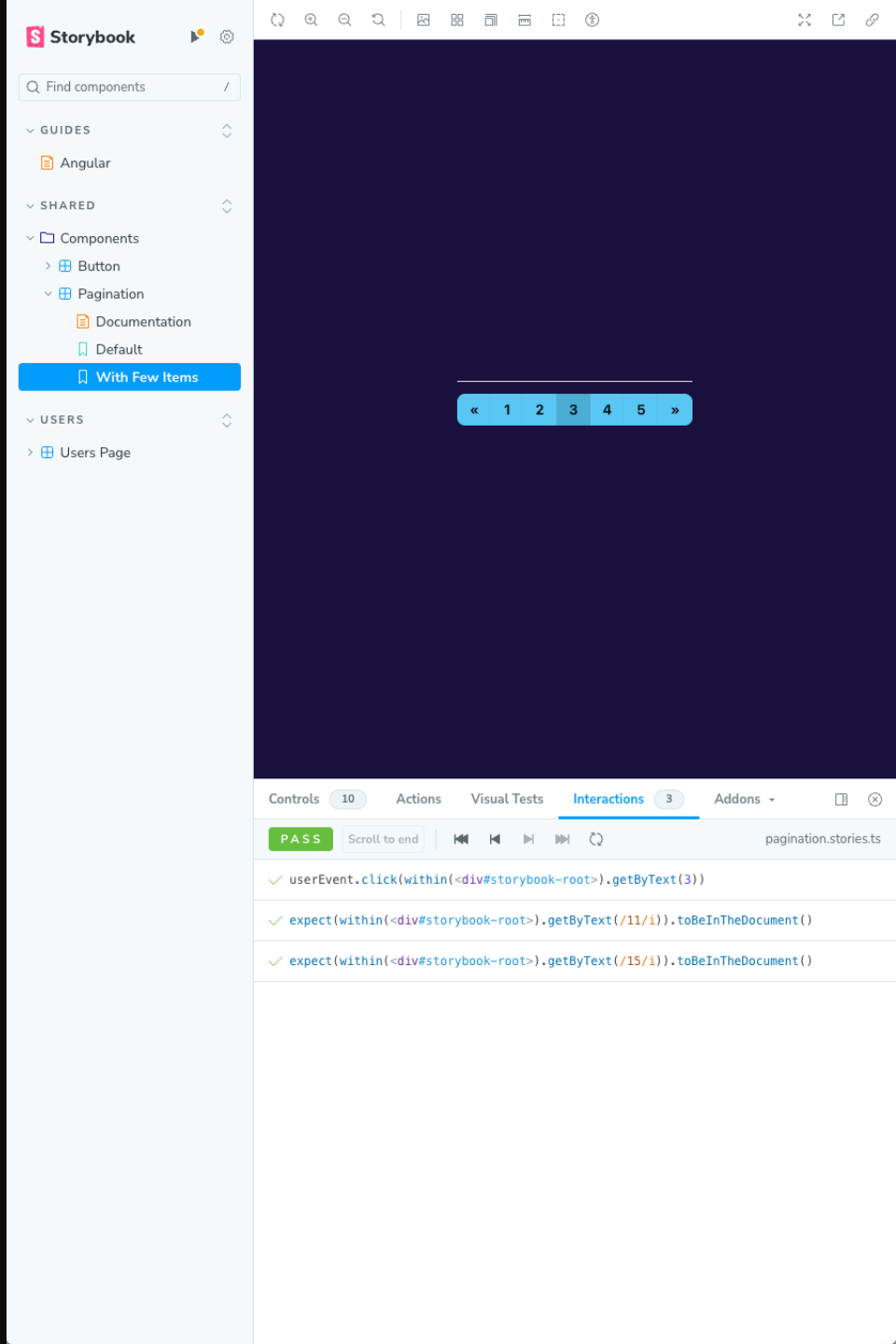
Pagination.stories.ts

```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,
  },
};
```



Pagination.stories.ts

```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();
  },
};
```



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

5:00

5-addons

Develop

Document

Test