

# **STORYBOOK PSEUDO STATE ADD-ON WEB WORKER SAAR MEETUP**

# AGENDA

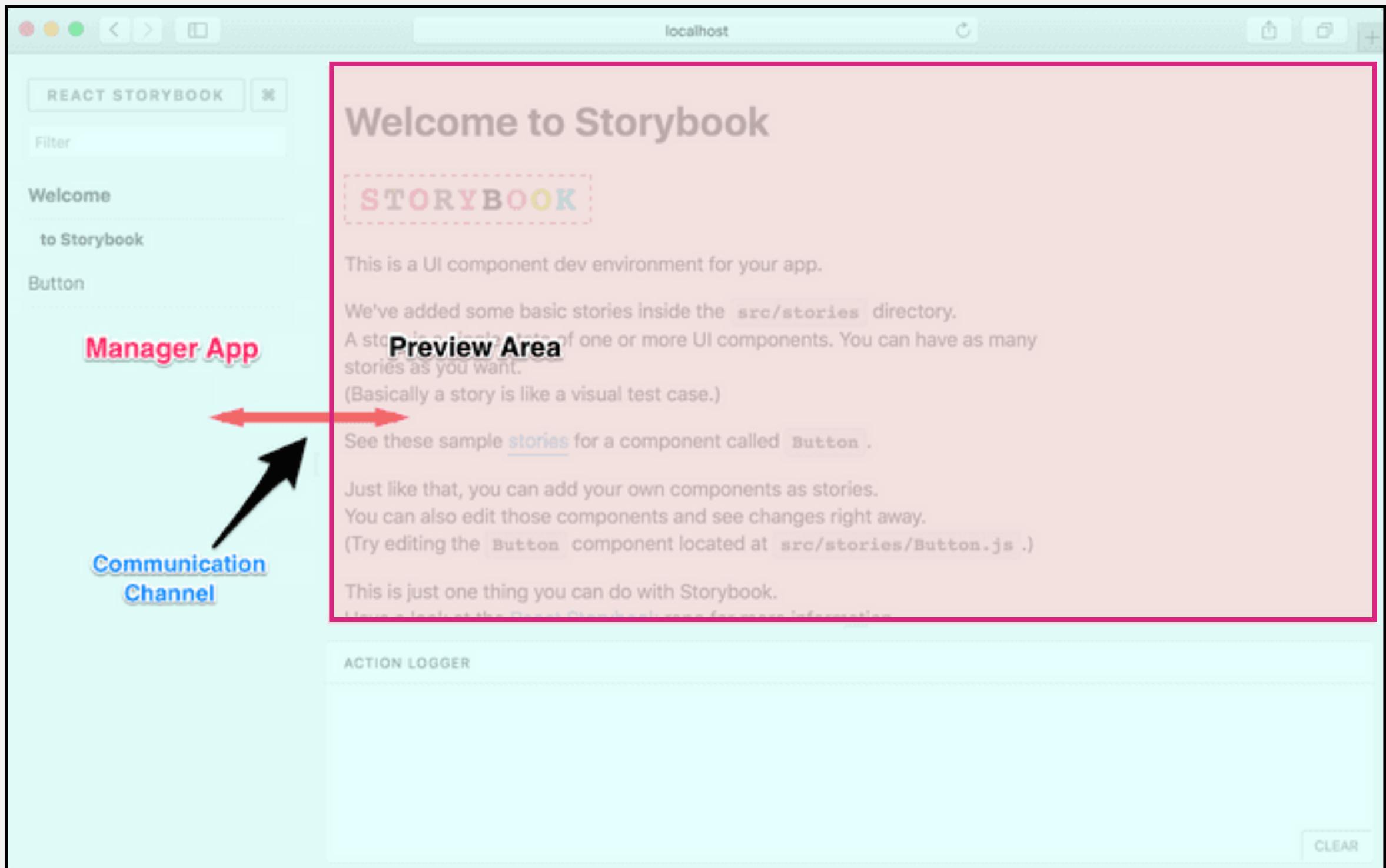
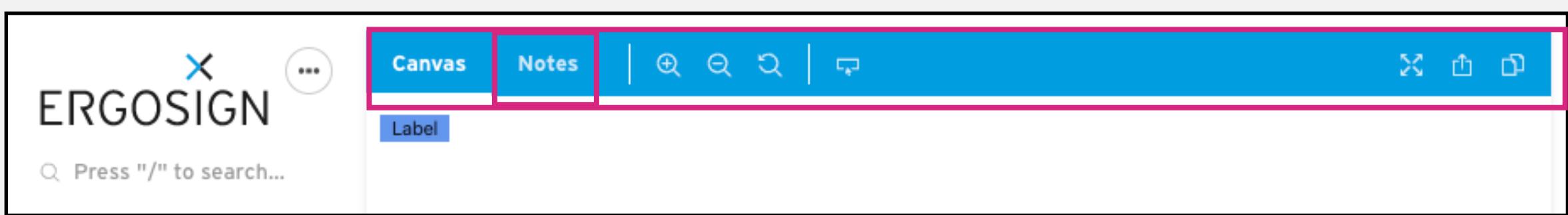
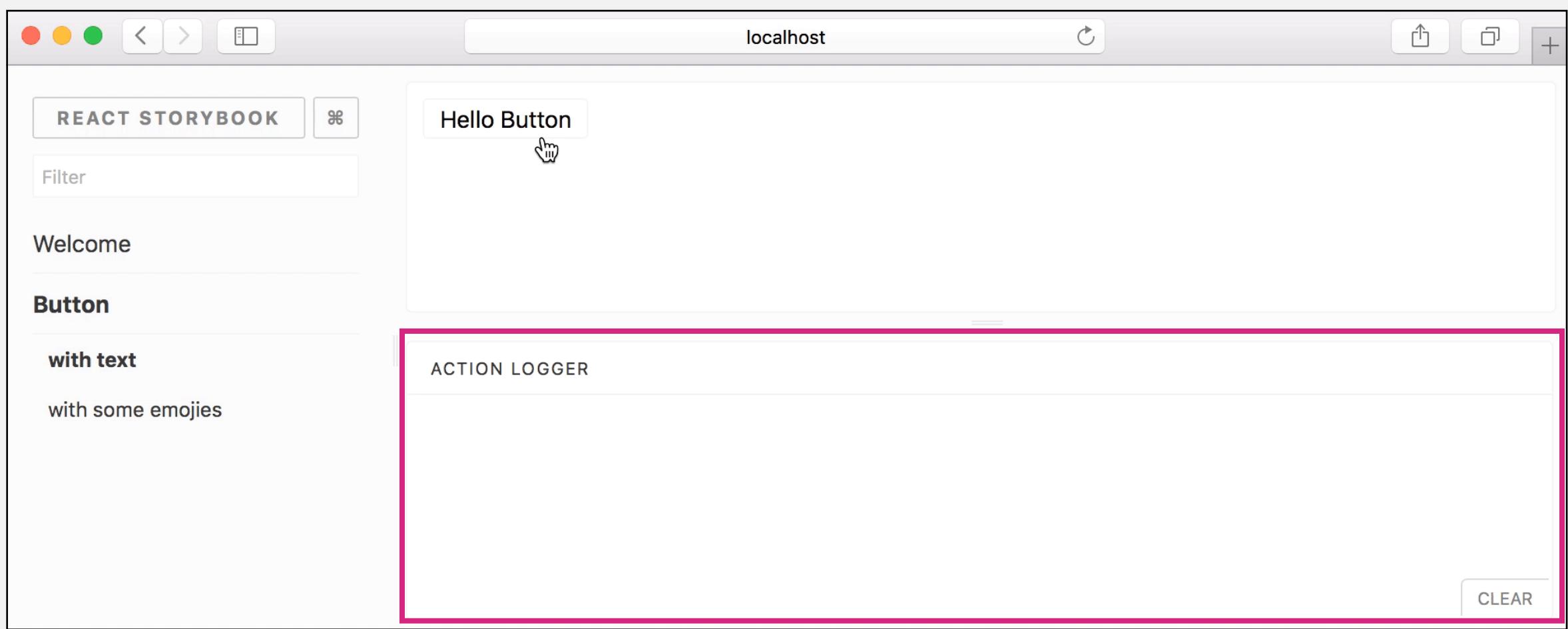
- > **Introduction**
- > **Storybook Add-on in General**
- > **Pseudo States Add-on**
  - > **Requirements**
  - > **Motivation**
- > **UX Library**
- > **Storybook Add-on**
- > **Technical Challenges**
- > **Example**



# PHILIPP SCHARDT

- › B.Sc. und M.Sc. Medieninformatik,  
Universität des Saarlandes
  - › Seit 2017 UX Software Engineer  
bei Ergosign
  - › Seit 2019 Technology Expert Web  
bei Ergosign
- 
- + Angular
  - + Typescript
  - + Web Components





# STORYBOOK ADD-ONS

## CAPABILITIES

- **Panel** (like Action Logger)
- **Tool** (like Zoom or Grid)
- **Tab** (like Notes)
- **Preview** (like Viewport)
  
- Interact/communicate with the preview/manager
- Interact/communicate with other add-ons
- Change storybook's state using it's APIs
- Navigating
  
- Addon Gallery

```
const styles = {
  textAlign: 'center',
};

const Center = ({ children }) =>
<div style={styles}>{children}</div>

// Story:
export default {
  title: 'Button',
};

export const defaultView = () => (
  <Center>
    <Button>Hello Button</Button>
  </Center>
);
```

# STORYBOOK ADD-ONS

## USAGE

- Components that wrap a story
- Decorators = wrapper component
- Native add-ons use Storybook as a platform and interact with it
- Documentation

```
import Button from './button';
import Center from './center';

export default {
  title: 'Button',
  decorators: [storyFn =>
    <Center>{storyFn()}</Center>
  ];
}

export const defaultView = () => (
  <Button>Hello Button</Button>
);
```

# STORYBOOK ADD-ONS

## USAGE

- Components that wrap a story
- **Decorators** = wrapper component
- Native add-ons use Storybook as a platform and interact with it
- [Documentation](#)



# PSEUDO STATES ADD-ON

## REQUIREMENTS

- Visualise **CSS Pseudo States**
  - :hover, :focus, :active
- Visualise **HTML Attributes**
  - :disabled, :readonly, :checked
- Visualise **Properties / Custom Attributes**
  - error, selected, open



# PSEUDO STATES ADD-ON

## MOTIVATION

- Increase Developer Experience (DX)
- Show style permutations of components
- Connect design and development
- Design Review
- Quality Improvement & Documentation
  - “Visual Unit Test”
- Framework agnostic

## Buttons

Button

Disclosure Button

Icon Label Button

Scrollspy Button

## Controls

Card

Detail Summary

Global Search Input

Image Hint

Info Area

Label Value Container

List Element

Loading Indicator

Logo

Message Toaster

Price Breakdown Table

Progress Bar

Progress Bar Button

Quantity Table

User Information

## Form Controls

Checkbox

Dropdown

Input

Input Autoresolution

Radio Button

## Normal Primary (Dark)

normal

hover

active

focus (normal)

disabled

&gt; HTML

## Normal Primary (Dark) Stretched

normal

hover

active

focus (normal)

disabled

&gt; HTML

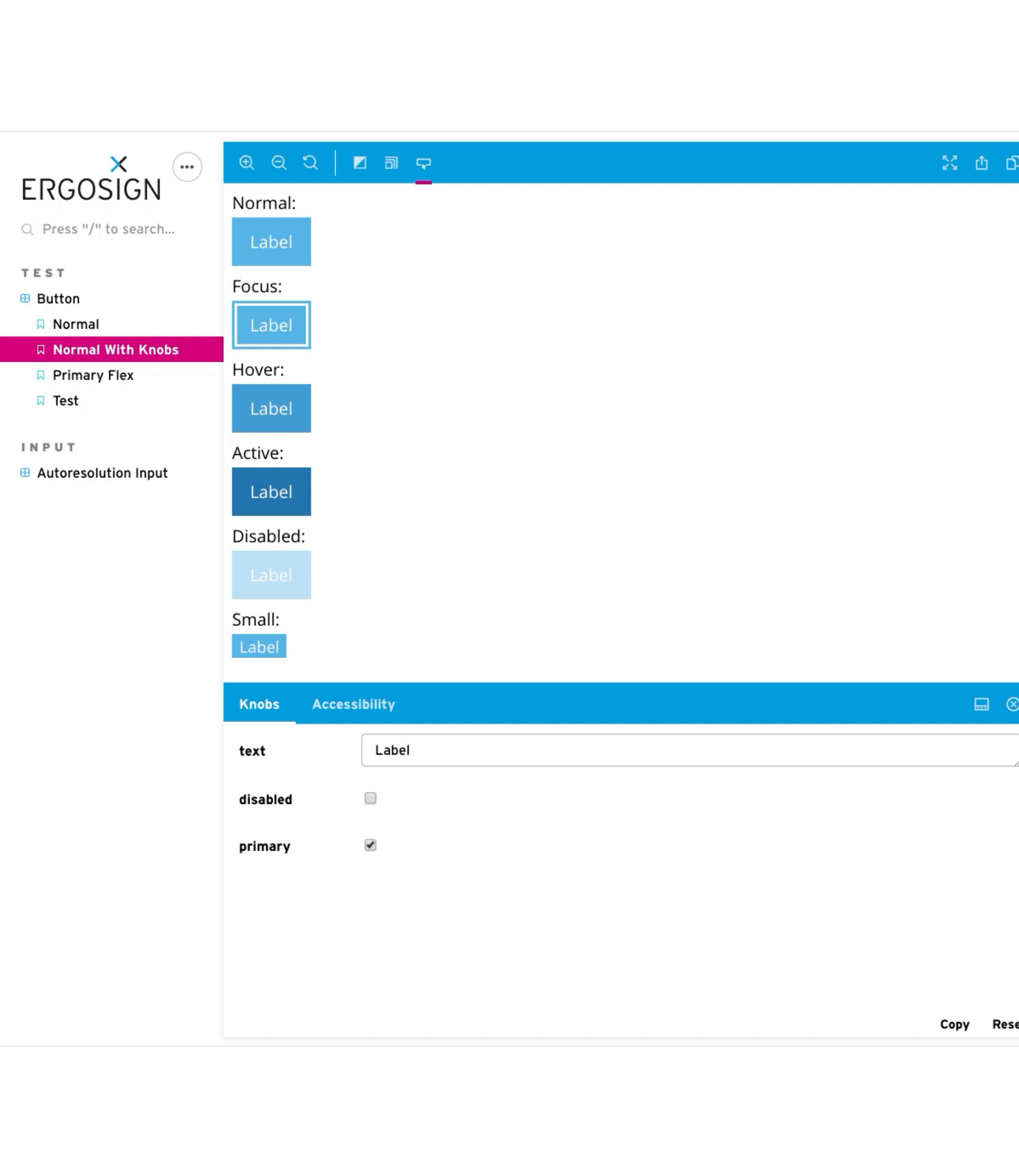
## UX LIBRARY

## BACKGROUND STORY

- > Storybook like solution by Ergosign ([Github Repo](#))
- > Generate components automatically
- > SASS Comment || JSDoc Comment
- > Parsed + Assembled

## &gt; Limitations

- > Only Plain HTML, AngularJs, Angular+ Components
- > New “Story” for each property
- > Manual generation of pseudo-styling  
:active, .active-pseudo { ... }
- > Hard to set up complex examples
  - > Reactive forms, example data (objects, arrays)
- > Maintenance overhead



# PSEUDO STATES ADD-ON

V0.0.1-ALPHA.19

- [Open Source \(Github Repo\)](#)
- Extends Storybook functionality
  - Decorator to extend story (withPseudo)
  - Show/Hide States (Toolbar Button)
- Frameworks
  - Angular
  - React
  - Vue
  - HTML
  - Web Components (Lit-HTML)

```
src > components > controls > three-canvas > three-canvas.utils.ts > <function>
11     sceneRenderer: Renderer,
12     cameraManager?: CameraManager
13   ): void {
14     if (mount && cameraManager && sceneRenderer) {
15       cameraManager.updateRatio(mount.offsetWidth / mount.offsetHeight);
16       sceneRenderer.setSize(mount.offsetWidth, mount.offsetHeight);
17     }
18   }
19
20   export function findSelectableObject(): void {
21
22     export function onDocumentMouseDown(
23       event: React.MouseEvent<HTMLDivElement, MouseEvent>,
24       mount: HTMLDivElement,
25       cameraManager: CameraManager,
26       sceneManager: SceneManager,
27       onSelectCallback?: (object?: Object3D) => void
28     ): void {
29       event.preventDefault();
30       const mouse3D = new Vector2(
31         ((event.clientX - mount.offsetLeft) / mount.clientWidth) * 2 - 1,
32         -((event.clientY - mount.offsetTop) / mount.clientHeight) * 2 + 1
33       );
34       const raycaster = new Raycaster();
35       raycaster.setFromCamera(mouse3D, cameraManager.currentCamera);
36       // @ts-ignore
37       // Can be ignored, because readonly does well for that function too, as it will not change it internally
38       const intersects = raycaster.intersectObjects(sceneManager.selectableObjects, true);
39
40       if (onSelectCallback) {
41         let selectedObject: Object3D | undefined = undefined;
42         const findSelectableObject = (ancestor: Object3D): void => {
43           if (isSelectable(ancestor) && selectedObject == null) {
44             selectedObject = ancestor;
45           }
46         };
47         for (let index = 0; index < intersects.length; index++) {
48           const hitObject = intersects[index].object;
49           if (isSelectable(hitObject)) {
50             selectedObject = hitObject;
51             break;
52           } else {
53             hitObject.traverseAncestors(findSelectableObject);
54             if (selectedObject) break;
55           }
56         }
57       }
58     }
59   }
60 }
```

# TECHNICAL CHALLENGES

- > Support different frameworks
    - > Mono Repo ([Lerna](#))
    - > NPM Package for each framework
    - > Shared code base (types, interfaces)
  - > Display story multiple times independently
    - > Framework dependent
  - > Share Show/Hide-State across stories
    - > ToggleButton (Type . Tool)
    - > Session Storage vs `useAddonState()` Hook ([Issue](#))
    - > Channel Communication: Tool <-> Decorator/Story

```
src > components > controls > three-canvas > three-canvas.utils.ts > <function>
11     sceneRenderer: Renderer,
12     cameraManager?: CameraManager
13   ): void {
14     if (mount && cameraManager && sceneRenderer) {
15       cameraManager.updateRatio(mount.offsetWidth / mount.offsetHeight);
16       sceneRenderer.setSize(mount.offsetWidth, mount.offsetHeight);
17     }
18   }
19
20   export function findSelectableObject(): void {
21
22     export function onDocumentMouseDown(
23       event: React.MouseEvent<HTMLDivElement, MouseEvent>,
24       mount: HTMLDivElement,
25       cameraManager: CameraManager,
26       sceneManager: SceneManager,
27       onSelectCallback?: (object?: Object3D) => void
28     ): void {
29       event.preventDefault();
30       const mouse3D = new Vector3(
31         ((event.clientX - mount.offsetLeft) / mount.clientWidth) * 2 - 1,
32         -((event.clientY - mount.offsetTop) / mount.clientHeight) * 2 + 1
33       );
34       const raycaster = new Raycaster();
35       raycaster.setFromCamera(mouse3D, cameraManager.currentCamera);
36       // @ts-ignore
37       // Can be ignored, because readonly does well for that function too, as it will not change it internally
38       const intersects = raycaster.intersectObjects(sceneManager.selectableObjects, true);
39
40       if (onSelectCallback) {
41         let selectedObject: Object3D | undefined = undefined;
42         const findSelectableObject = (ancestor: Object3D): void => {
43           if (isSelectable(ancestor) && selectedObject == null) {
44             selectedObject = ancestor;
45           }
46         };
47         for (let index = 0; index < intersects.length; index++) {
48           const hitObject = intersects[index].object;
49           if (isSelectable(hitObject)) {
50             selectedObject = hitObject;
51             break;
52           } else {
53             hitObject.traverseAncestors(findSelectableObject);
54             if (selectedObject) break;
55           }
56         }
57       }
58     }
59   }
60 }
```

# TECHNICAL CHALLENGES

- > Force component to be in html state (focus, hover, active)
  - > Automatically generated pseudo styling
    - > PostCSS
    - > PostCSS Pseudo Classes Plugin
    - > Preset (extends webpack' sass rule: postcss-loader)
  - > Add generated styling class to component story
    - > Scoped component styling (css-modules, View-Encapsulation, shadow-dom, Scoped CSS)
    - > Find element with interaction styling (selection)
    - > Add class after story is rendered

```

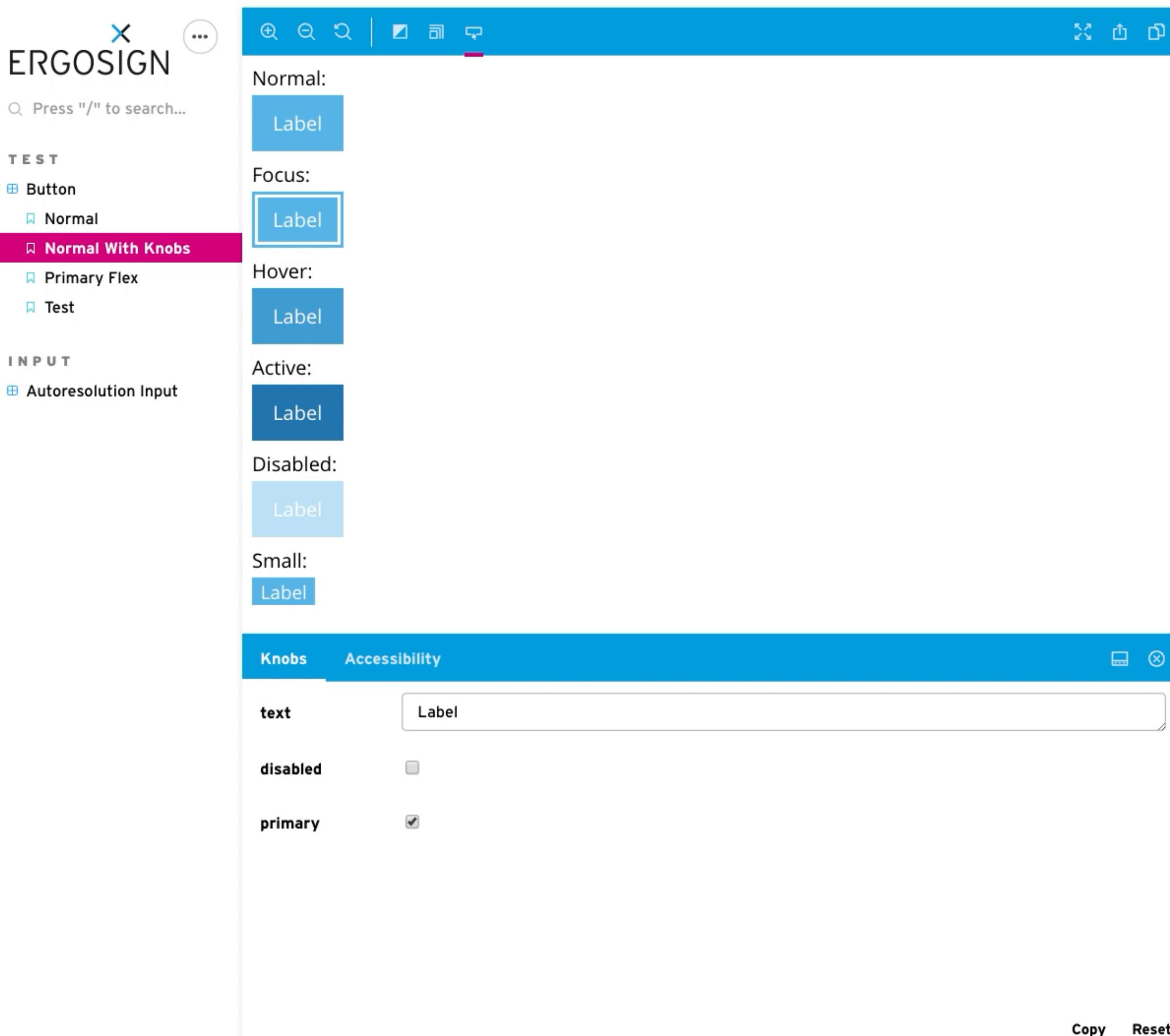
wall-resizing.ts      three-canvas.tsx    index.d.ts    three-canvas.utils.ts  selection-behavior.ts
src > components > controls > three-canvas > three-canvas.utils.ts > <function>
  11   sceneRenderer: Renderer,
  12   cameraManager?: CameraManager
  13 ): void {
  14     if (mount && cameraManager && sceneRenderer) {
  15       cameraManager.updateRatio(mount.offsetWidth / mount.offsetHeight);
  16       sceneRenderer.setSize(mount.offsetWidth, mount.offsetHeight);
  17     }
  18   }
  19
  20   export function findSelectableObject(): Object3D | undefined {
  21
  22     export function onDocumentMouseDown(
  23       event: React.MouseEvent<HTMLDivElement, MouseEvent>,
  24       mount: HTMLDivElement,
  25       cameraManager: CameraManager,
  26       sceneManager: SceneManager,
  27       onSelectCallback?: (object?: Object3D) => void
  28     ): void {
  29       event.preventDefault();
  30       const mouse3D = new Vector3();
  31       ((event.clientX - mount.offsetLeft) / mount.clientWidth) * 2 - 1,
  32       -((event.clientY - mount.offsetTop) / mount.clientHeight) * 2 + 1
  33     );
  34     const raycaster = new Raycaster();
  35     raycaster.setFromCamera(mouse3D, cameraManager.currentCamera);
  36     // @ts-ignore
  37     // Can be ignored, because readonly does well for that function too, as it will not change it internally
  38     const intersects = raycaster.intersectObjects(sceneManager.selectableObjects, true);
  39
  40     if (onSelectCallback) {
  41       let selectedObject: Object3D | undefined = undefined;
  42       const findSelectableObject = (ancestor: Object3D): void => {
  43         if (isSelectable(ancestor) && selectedObject == null) {
  44           selectedObject = ancestor;
  45         }
  46       };
  47       for (let index = 0; index < intersects.length; index++) {
  48         const hitObject = intersects[index].object;
  49         if (isSelectable(hitObject)) {
  50           selectedObject = hitObject;
  51           break;
  52         } else {
  53           hitObject.traverseAncestors(findSelectableObject);
  54           if (selectedObject) break;
  55         }
  56       }
  
```

# TECHNICAL CHALLENGES

- Force property of component to be enabled
- Angular's change detection
- Only access to story representation of component
- In React:
 

```
const alteredStory = {
  ...story,
  props: {
    ...story.props,
    [property]: true
  },
};
```
- Testing

# EXAMPLE



```
export default section = {  
  component: ButtonComponent,  
  title: "Test|Button",  
  moduleMetadata: {  
    declarations: [ ButtonComponent ],  
    imports: [ CommonModule ]  
  },  
  decorators: [  
    withPseudo  
  ],  
  parameters: {  
    withPseudo: {  
      selector: 'button',  
      pseudos: PseudoStatesDefault,  
      attributes: ['disabled', 'small']  
    } } };  
  
export const Normal = () => { . . . };
```

**THANK  
YOU!**

**ERGOSIGN**  


BERLIN

HAMBURG

MÜNCHEN

**SAARBRÜCKEN**

STUTTGART

ZÜRICH