# Storybook

# Storybook Overview

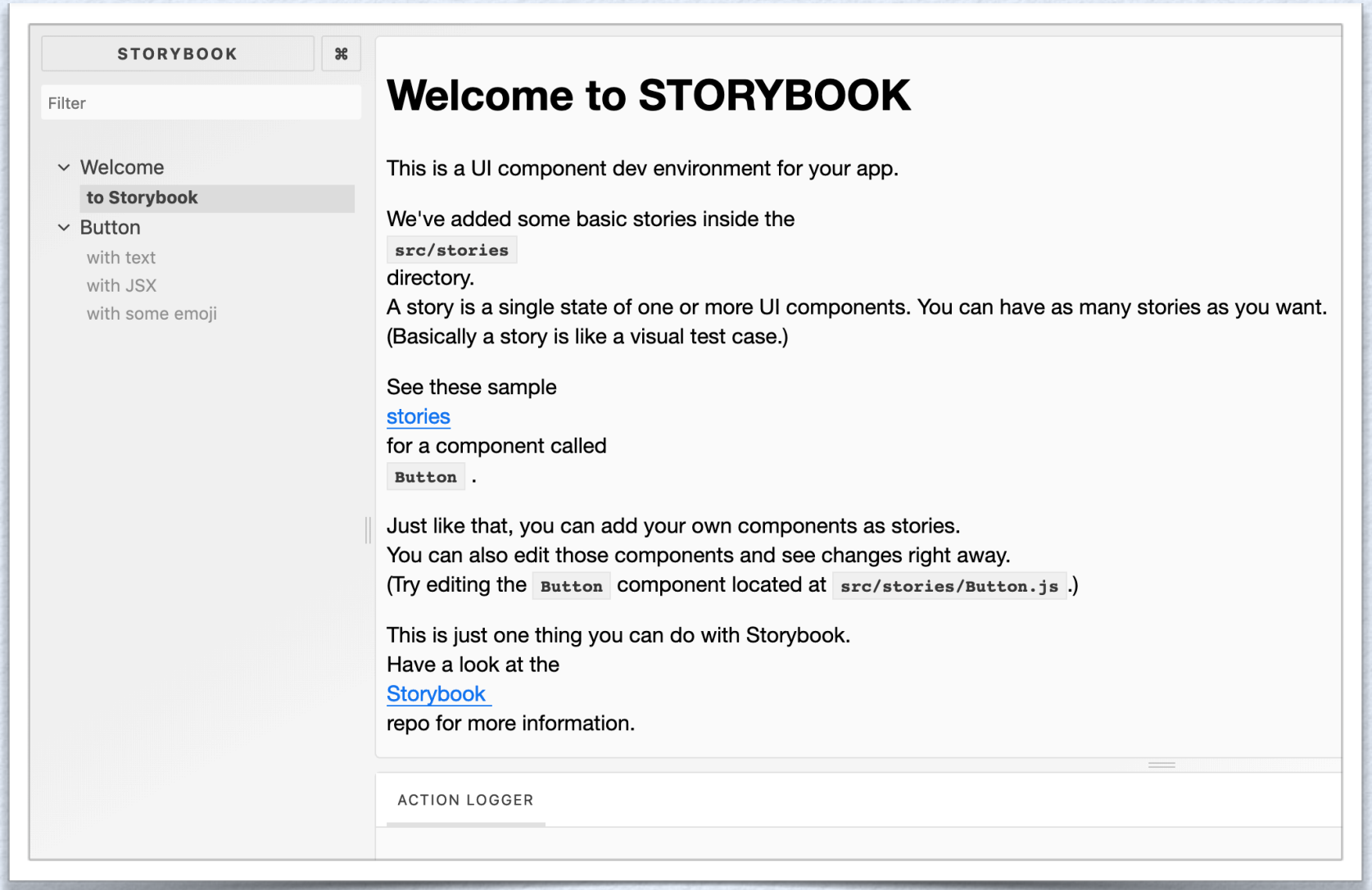
- "A UI development environment and playground for UI components"
- "Showcase components interactively in an isolated development environment"
- Works with many web UI frameworks including React, Vue, and Angular
- Benefits
  - can implement and test components in isolation before pages that will use them are implemented
  - designers can experiment with component styling without running the app
  - can demonstrate components to customers for feedback before app is complete
  - can verify operation of components without following a long navigation sequence in the app
  - can use "design system" documentation

# Installing Storybook

- **`npx -p @storybook/cli sb init`** storybook initialize
- Adds **`storybook`** and **`build-storybook`** npm scripts in **`package.json`**
- Creates **`.storybook`** directory containing
  - **`config.js`**
    - modify **`config.js`** to use plugins like Vuex
    - modify to register custom components (Some components don't need this! Which do?)
  - **`addons.js`**
- Creates **`stories`** directory containing
  - **`index.stories.js`** - defines the stories
  - **`Welcome.js`** - an example component that provides instructions on how to add stories
  - **`MyButton.js`** - another example component

# Running Storybook

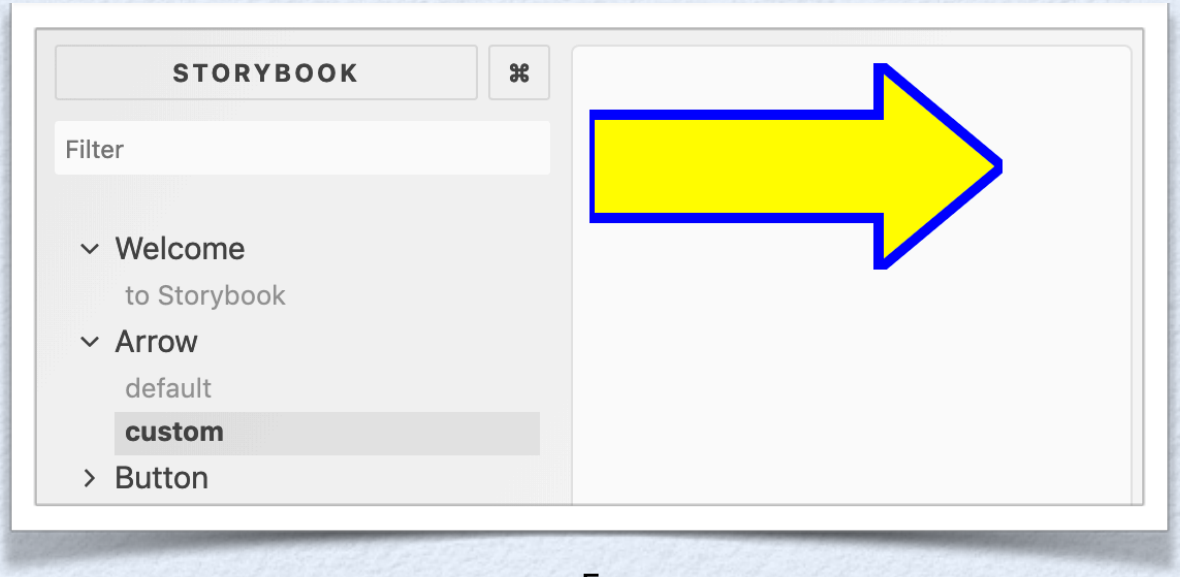
- `npm run storybook`



# Adding Components

- Arrow component example

```
storiesOf('Arrow', module)
  .add('default', () => ({
    components: {Arrow},
    template: '<Arrow :height="100" :width="200" />'
  }))
  .add('custom', () => ({
    components: {Arrow},
    template:
      '<Arrow fill="yellow" :height="100" stroke="blue" :strokeWidth="5" :width="200" />'
  }));
```



# Default CSS

- To provide default styling defined outside components being rendered
  - create **`stories/storybook.css`**

```
body {
  font-family: sans-serif;
}
```

  - import near top of **`index.stories.js`**

```
import './storybook.css';
```

# Mocking Fetch

- Want ability to demonstrate components that make REST calls without actually making the calls
- When using Fetch API, one way is to replace **`fetch`** function with a no-op in **`.storybook/config.js`**

```
// Replace the fetch function with a no-op
// so components that make REST calls don't fail.
window.fetch = () => {};
```

# More Examples

```
storiesOf('Scale', module).add('default', () => ({
  store,
  components: {Scale},
  methods: {
    onChange(value) {
      console.info('Scale value is', value);
    }
  },
  template: '<Scale :onchange="onChange" path="scaleDemo" />'
}));
```

```
storiesOf('TemperatureBar', module).add('default', () => ({
  store,
  components: {TemperatureBar},
  data() {
    return {temperature: 60};
  },
  template:  template: `<TemperatureBar
    title="Temperature"
    dimension="temperature"
    start="Cold"
    end="Hot"
    :goal=60
    :value="temperature"
  />`
}));
```

# Add-ons

- Add functionality to Storybook
- Documented at https://storybook.js.org/addons/addon-gallery/
- Some only work with certain web frameworks
- Some are maintained by the Storybook team
  and others are maintained by the community
- Steps to install each differ, so check documentation