

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, Angular, and Svelte
- 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 as “design system” documentation
- <https://storybook.js.org/>

Installing Storybook for Vue

- `npx -p @storybook/cli sb init --type vue`
- 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`
 - imports addons to display data received by Storybook event handlers and create links that navigate between stories
- Creates `stories` directory containing
 - `index.stories.js` - defines stories
 - it's also possible to colocate story definitions with component implementations under `src` directory
 - `Welcome.js` - an example component that provides instructions on how to add stories
 - `MyButton.js` - another example component

Running Storybook

- `npm run storybook`

The screenshot shows the Storybook application interface. At the top, four boxes labeled 'zooms out', 'zooms in', 'resets zoom', and 'toggles grid' have arrows pointing to their respective icons in the top toolbar. The left sidebar contains a search bar and a list of components: 'Welcome', 'to Storybook' (highlighted), 'Button', 'with text', 'with JSX', and 'with some emoji'. A text box at the bottom of the sidebar states: 'Stories for **Welcome** and **Button** components are supplied out of the box. Replace them with stories for your components.' The main content area displays 'Welcome to STORYBOOK' and provides information about the UI component dev environment, including the `src/stories` directory and a definition of a story. The bottom of the interface shows a tab labeled 'Actions'.

zooms out zooms in resets zoom toggles grid

• `npm run storybook`

Storybook

Q Press "/" to search...

Welcome

to Storybook

Button

with text

with JSX

with some emoji

Stories for **Welcome** and **Button** components are supplied out of the box. Replace them with stories for your components.

Welcome to STORYBOOK

This is a UI component dev environment for your app.

We've added some basic stories inside the `src/stories` directory.

A story is a single state of one or more UI components. You can have as many stories as you want. (Basically a story is like a visual test case.)

See these sample [stories](#) for a component called `Button`.

Just like that, you can add your own components as stories.

[Actions](#)

Adding Stories ...

- In `stories/index.stories.js`

```
import {storiesOf} from '@storybook/vue';  
import {action} from '@storybook/addon-actions';  
  
import Arrow from '../src/components/Arrow';  
import Todo from '../src/components/Todo';  
import TodoList from '../src/components/TodoList';
```

`action` function is for
mocking event handling
and logging it;
see examples in story
for `Todo` component

stories for imported components
are on next three slides

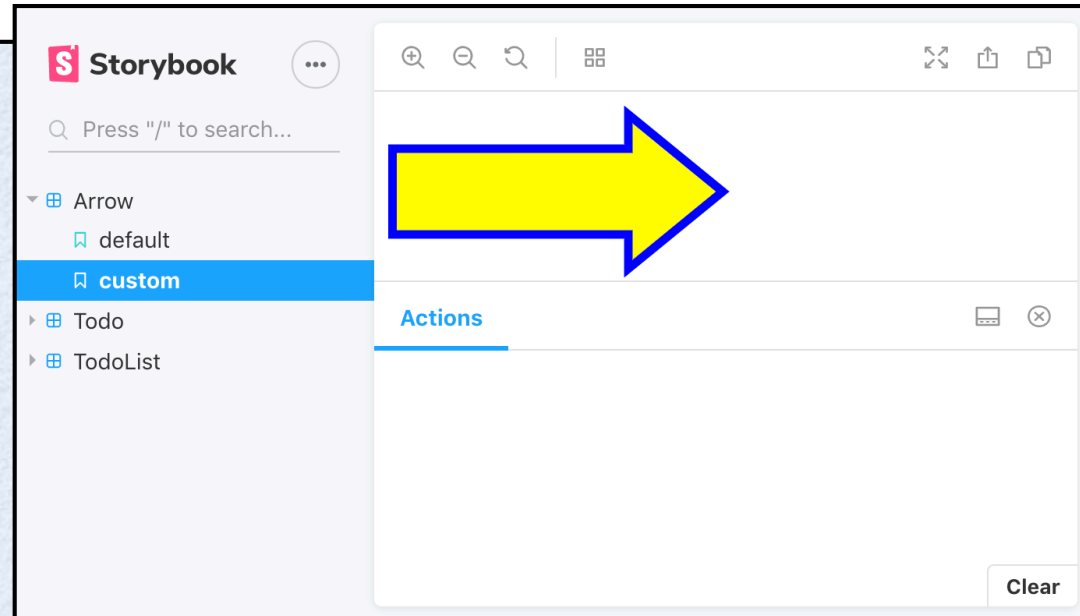
... Adding Stories ...

- **Arrow** component stories

```
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" />'
  }));
```

Passing `module` is necessary for **"hot module replacement"**.

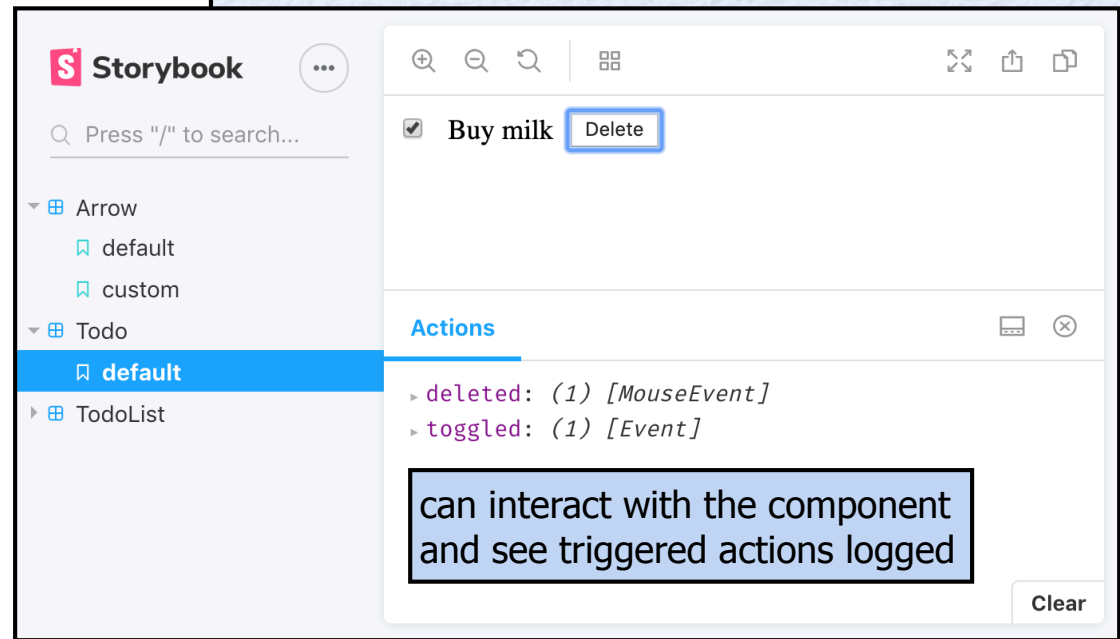
Webpack uses this to add and remove modules at runtime without a full reload. This way everything needed by every component doesn't need to be retained in memory the entire time Storybook is running.



... Adding Stories ...

- **Todo** component story

```
storiesOf('Todo', module).add('default', () => {  
  return {  
    components: {Todo},  
    data() {  
      return {  
        todo: {text: 'Buy milk'}  
      };  
    },  
    template: `<Todo  
      :todo="this.todo"  
      :onDeleteTodo="this.deleteAction"  
      :onToggleDone="this.toggleAction"  
    />`,  
    methods: {  
      deleteAction: action('deleted'),  
      toggleAction: action('toggled')  
    }  
  };  
});
```



... Adding Stories

- **TodoList** component story

```
storiesOf('TodoList', module).add('default', () => {  
  return {  
    components: {TodoList},  
    template: '<TodoList />'  
  };  
});
```

The screenshot displays the Storybook application interface. On the left, a sidebar lists the component hierarchy: 'Arrow' (with 'default' and 'custom' stories), 'Todo' (with 'default' story), and 'TodoList' (with 'default' story selected). The main area shows the 'To Do List' story. It features a title 'To Do List', a subtitle '1 of 2 remaining', and a button 'Archive Completed'. Below this is an input field 'enter new todo here' with an 'Add' button. A list of todos is shown: 'learn Vue' (checked) and 'build a Vue app' (unchecked), each with a 'Delete' button. At the bottom, there is an 'Actions' panel and a 'Clear' button. A blue callout box with the text 'can do everything here that todo app can do' is overlaid on the right side of the main area.

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

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


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

More

- There's more to learn about Storybook, but we've covered the basics
- For more read the official docs starting at <https://storybook.js.org/docs/basics/introduction/>