

## Product Design

# Building a product component library and design system with Storybook

May 30, 2019



Niall Maher

Founder of Codú Software Solutions

*This tutorial assumes you have some knowledge of modern JavaScript and [Node.js](#) installed on your system.*

Storybook is a tool used by companies like *Airbnb*, *Lyft* and *Salesforce* to develop, test and document their component libraries and design systems.

It's available for all the major view layer including libraries/frameworks (React, Vue, Angular, React Native, and Ember and more).

Storybook is a development sandbox that uses the concept of **stories** to document, test and build components.

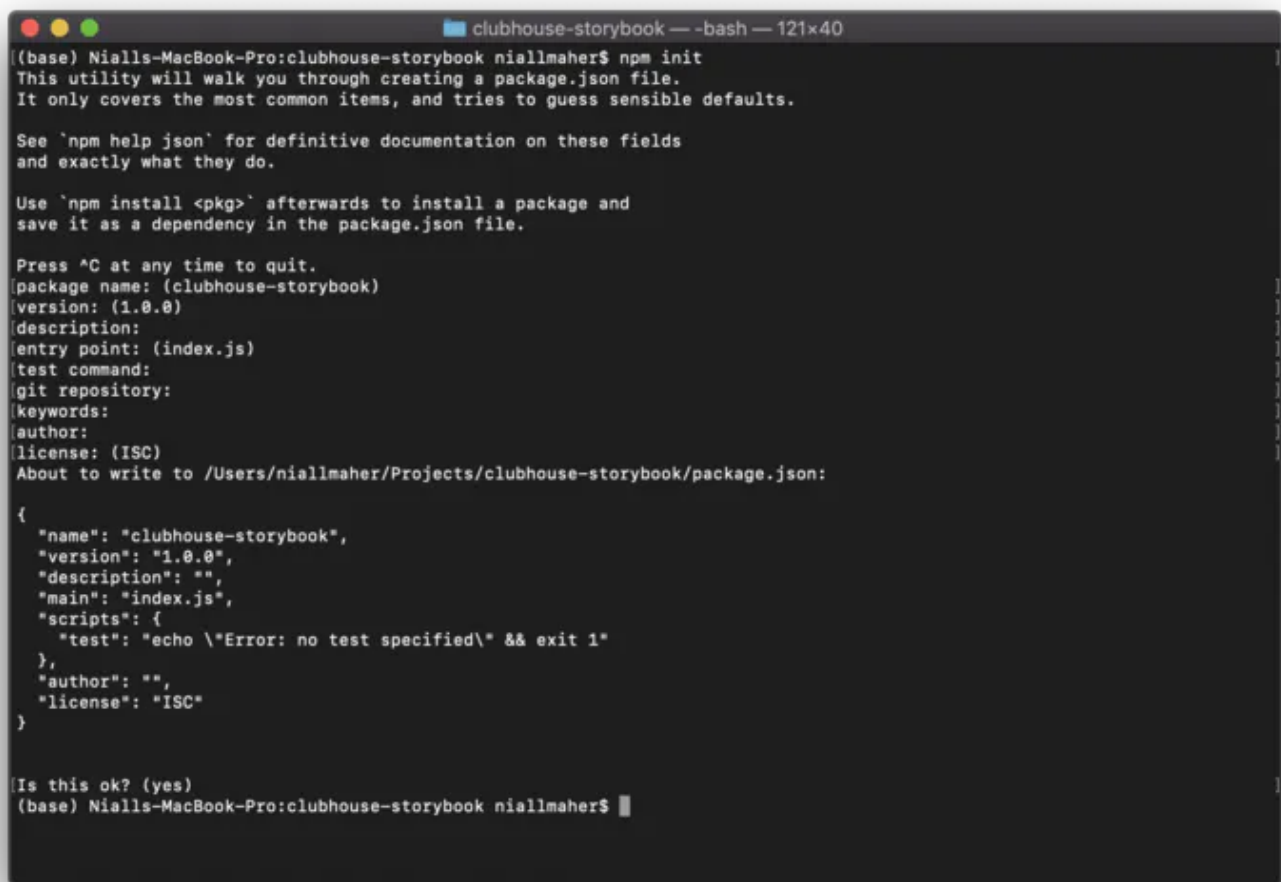
A story usually contains a single state of one component, almost like a visual test case. Technically a story is a function that returns something that can be rendered to the screen.

# Getting started

Let's start a new project, I've created a folder called "*clubhouse-storybook*", you can name yours whatever you like. Open your terminal in this directory and let's get started!

The first thing we need to do is make sure we have a *package.json* in our root directory.

Run `npm init` and just hit enter through all the defaults options.



```
clubhouse-storybook — -bash — 121x40
(base) Nialls-MacBook-Pro:clubhouse-storybook niallmaher$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See 'npm help json' for definitive documentation on these fields
and exactly what they do.

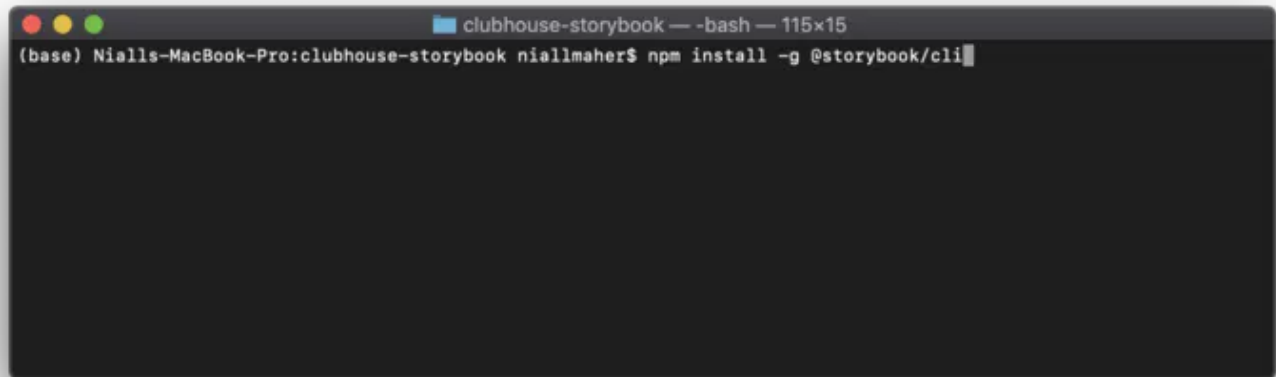
Use 'npm install <pkg>' afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
(package name: (clubhouse-storybook)
(version: (1.0.0)
(description:
(entry point: (index.js)
(test command:
(git repository:
[keywords:
(author:
(license: (ISC)
About to write to /Users/niallmaher/Projects/clubhouse-storybook/package.json:
{
  "name": "clubhouse-storybook",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

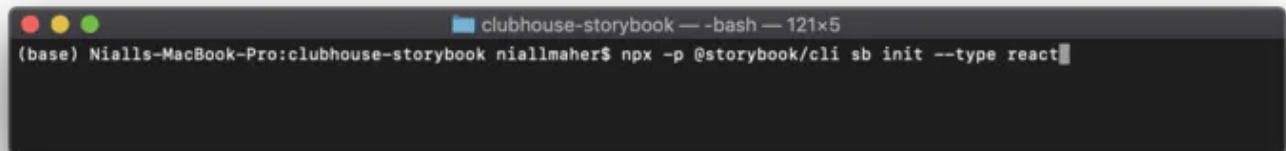
Is this ok? (yes)
(base) Nialls-MacBook-Pro:clubhouse-storybook niallmaher$
```

Storybook comes with a handy command line interface tool that makes getting a project started super fast. We need to install the storybook CLI tools globally and we do this by running:

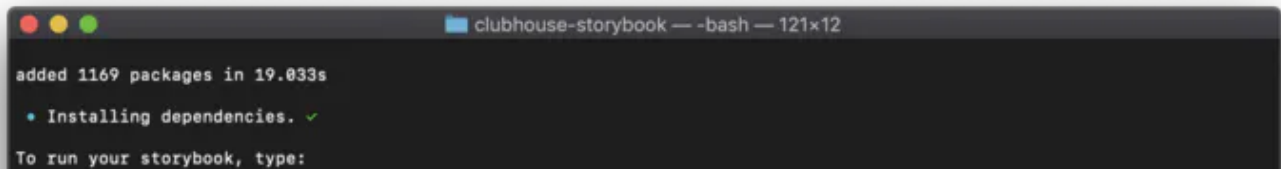
```
npm i -g @storybook/cli
```



Once the CLI is installed we have to initiate the app, I'm going to be using React for this tutorial but feel free to choose a different flavour. To initiate a project with React we run `npx -p @storybook/cli sb init --type react`.



This takes a few seconds as it bootstraps our project into a development-ready state.

A terminal window titled 'clubhouse-storybook — -bash — 121x12' showing the output of a command. The text in the terminal is: 'added 1169 packages in 19.033s', '• Installing dependencies. ✓', and 'To run your storybook, type:'.

```
clubhouse-storybook — -bash — 121x12
added 1169 packages in 19.033s
• Installing dependencies. ✓
To run your storybook, type:
```

Clubhouse! 



### Note for adding storybook to an existing project:

If you are running storybook in an existing app with a library chosen you can use `npx -p @storybook/cli sb init` which scans your package.json dependencies and installs the dependencies to run for your chosen view layer.

## Development

Once you have the previous steps completed it's simple to run the development server, just run `npm run storybook`.

```
clubhouse-storybook — open • npm TERM_PROGRAM=Apple_Terminal NVM_CD_FLAGS= SHE...
((base) Nialls-MacBook-Pro:clubhouse-storybook niallmaher$ npm run storybook

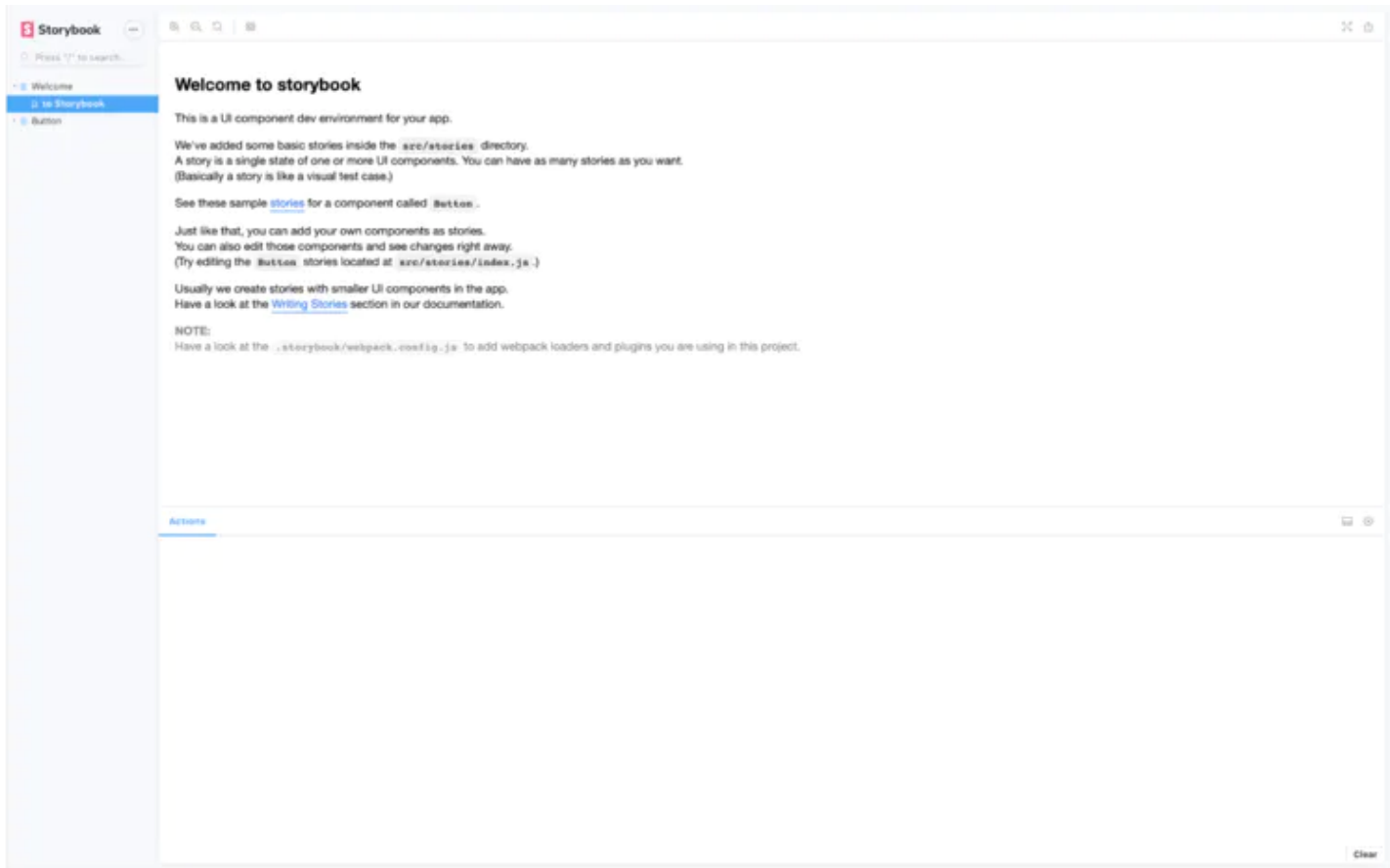
> clubhouse-storybook@1.0.0 storybook /Users/niallmaher/Projects/clubhouse-storybook
> start-storybook -p 6006

info @storybook/react v5.0.11
info
info => Loading presets
info => Loading presets
info => Loading custom addons config.
info => Using default webpack setup.
info => Using base config because react-scripts is not installed.
webpack built 094bac5507ddf4c984a5 in 5129ms

Storybook 5.0.11 started
6.5_s for manager and 6.26_s for preview

Local:      http://localhost:6006/
On your network: http://192.168.1.11:6006/
```

This by default will open your browser on <http://localhost:6006/> and should look something like this depending on the version:



Now it's time to crack open the code editor, finally!

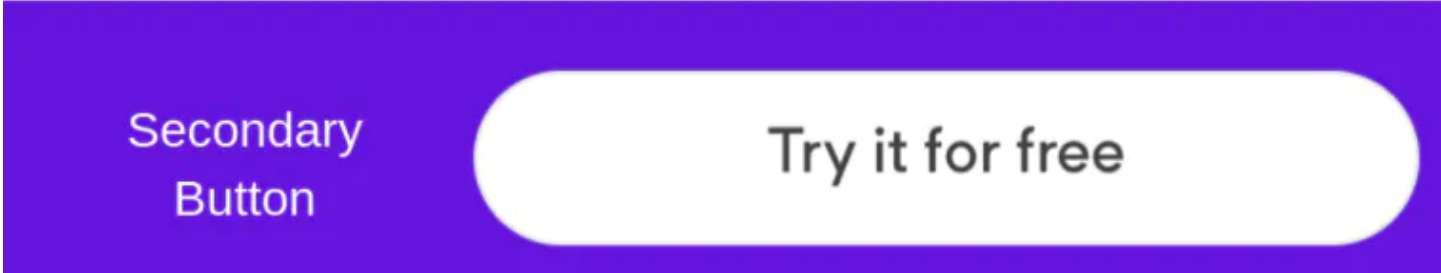
Take some time to familiarise yourself with the folders and files. You should notice, each component has it's own `.stories.js` extension, it's in these files that we write the "story" for each component and get it on the screen.

I'm going to use a simple button as a sample for the sample of how to develop a component with 2 variations, a primary and a secondary.

Here's a component a [top designer](#) sent me and I totally didn't rip it off the Clubhouse homepage 😊

A purple rounded rectangular button with the text "Try it for free" in white.

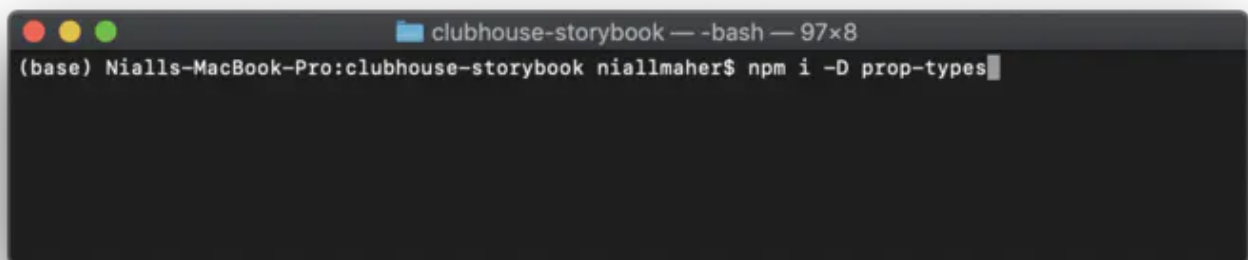
Primary  
Button

A purple rectangular bar containing a white rounded rectangular button with the text "Try it for free" in dark gray. To the left of the button is the text "Secondary Button" in white.

Secondary  
Button

Try it for free

One more package I like to use is the `prop-types` package, I find it helpful for when we document components (you'll see how soon). Install it with `npm i -D prop-types`.

A screenshot of a macOS terminal window. The title bar shows "clubhouse-storybook" and "bash" with window control buttons. The terminal text shows the command "npm i -D prop-types" being executed in a directory named "clubhouse-storybook".

```
clubhouse-storybook — bash — 97x8
(base) Nialls-MacBook-Pro:clubhouse-storybook niallmaher$ npm i -D prop-types
```

First, we will create a new folder called “Button” inside the “stories” folder, this folder will hold all the files to do with the button we are about to create. The next steps are my inside the folder we will create a `index.js` (to export the component), `Button.js`, `Button.stories.js` and `Button.css`.

Then so we can focus on the story writing copy and paste the following into the corresponding files.

## Button.js:

```
1  import React from 'react'
2  import PropTypes from 'prop-types'
3  import './Button.css'
4
5  const Button = ({ onClick, children, disabled, secondary, type }) => {
6
7    const className = 'c-Button'
8
9    const activeClassNames = [
10      className,
11      disabled ? `${className}--disabled` : '',
12      secondary ? `${className}--secondary` : '',
13    ].filter(Boolean).join(' ')
14
15    return (
16      <button
17        disabled={disabled}
18        tabIndex={0}
19        onClick={onClick}
20        type={type}
21        className={activeClassNames}
22      >
23        {children}
24      </button>
25    )
26  }
27
28  Button.propTypes = {
29    children: PropTypes.node.isRequired,
30    disabled: PropTypes.bool.isRequired,
31    secondary: PropTypes.bool.isRequired,
32    onClick: PropTypes.func.isRequired,
33    type: PropTypes.oneOf(['button', 'submit', 'reset']),
34  }
35
36  Button.defaultProps = {
37    children: null,
38    disabled: false,
39    secondary: false,
```



```
40   onClick: noop,  
41   type: 'button',  
42 }  
43  
44 export default Button
```

Button.js hosted with ❤ by GitHub

[view raw](#)

## Button.css:

```
1  .c-Button {  
2    background-color: #6515dd;  
3    border: 1px solid transparent;  
4    border-radius: 50px;  
5    box-shadow: none;  
6    box-sizing: border-box;  
7    color: #fff;  
8    cursor: pointer;  
9    display: flex;  
10   font-size: 18px;  
11   height: 50px;  
12   justify-content: center;  
13   min-width: 260px;  
14   outline: none;  
15   padding: 0 30px;  
16   transition: all 0.2s ease-in-out;  
17 }  
18 .c-Button:hover,  
19 .c-Button:focus {  
20   background-color: #5011AE;  
21 }  
22  
23 .c-Button--disabled {  
24   opacity: 0.5;  
25   user-select: none;  
26 }  
27 .c-Button--disabled:hover {  
28   cursor: not-allowed;  
29 }  
30 .c-Button--secondary {  
31   background-color: #fff;  
32   border: 1px solid #E4E4E4;  
33   color: #414042;
```

```
34   transition: all 0.2s ease-in-out;
35 }
36 .c-Button--secondary:hover,
37 .c-Button:focus {
38   background-color: #E6E6E6;
39 }
```

Button.css hosted with ❤ by GitHub

[view raw](#)

index.js:

```
1  import Button from './Button'
2  export default Button
```

index.js hosted with ❤ by GitHub

[view raw](#)

## Storytime!

Open your `Button.stories.js` file and import the following:

First, we import “React” since we are using JSX.

```
import React from 'react'
```

Then we import `storybook` which is a named export from `@storybook/react`.

```
import { storiesOf } from '@storybook/react'
```

Then `action` which is a named export from `@storybook/addon-actions`.

```
import { action } from '@storybook/addon-actions'
```

Then we import our `Button` component is a named export from `index.js`.

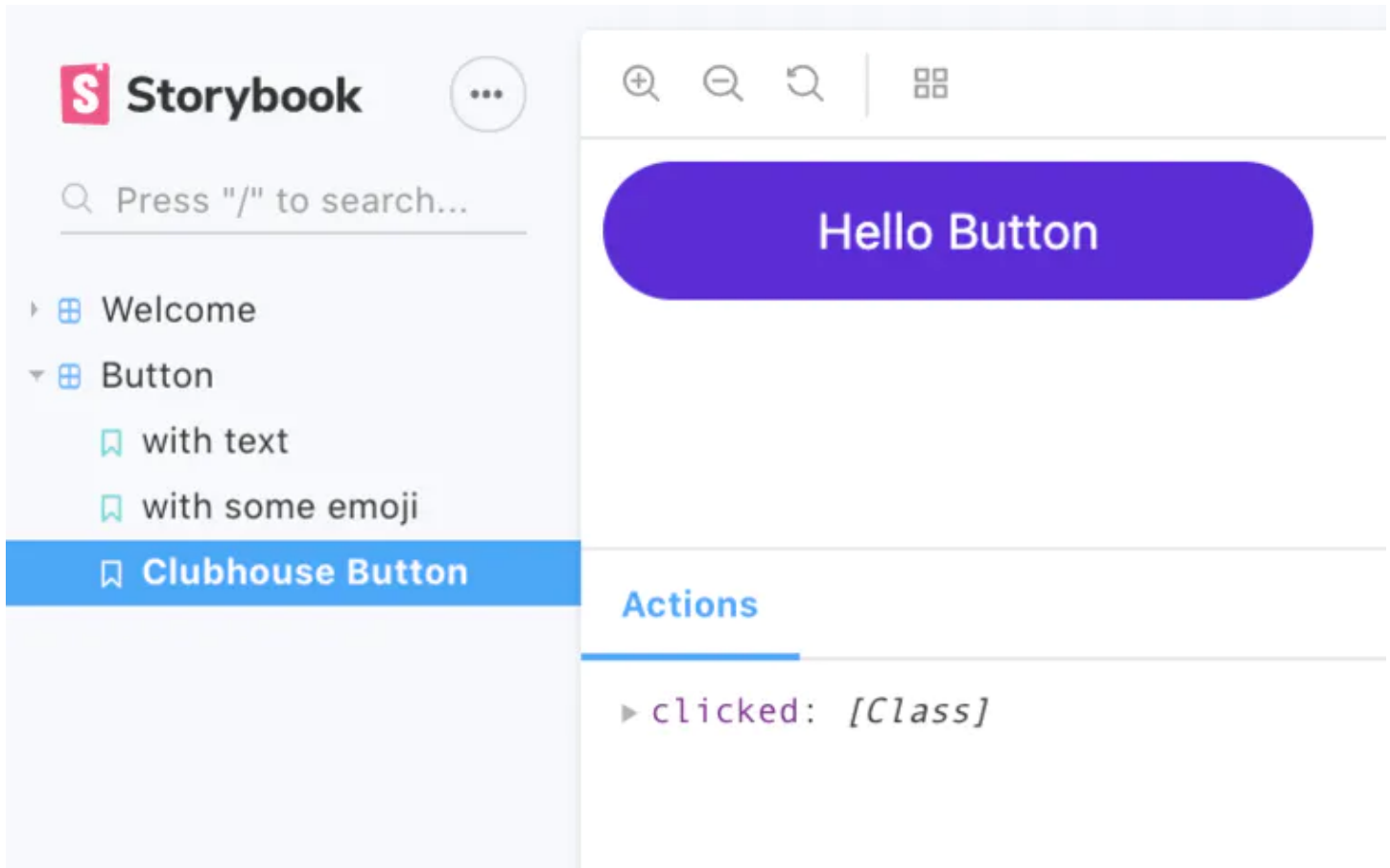
```
import Button from './'
```

To create a story we pass the group name of the component which is “Button”, this is what shows up in the sidebar of the storybook to organise the components, we then have to pass the “module” as a second parameter. Storybook needs a reference to the file/module where your story code is to enable `hpt-module-replacement`. If you do not supply it, you’d need to refresh your browser for every change you make to your component and story code.

Then we call the “add” function and pass the component name which is “Clubhouse Button” and the second parameter which is a function that returns our component. We also are going to pass the “action” addon to the “onClick” that we see used in the demo components.

```
storiesOf('Button', module)
  .add('Clubhouse Button', () => <Button onClick={action('c
```

You should now see a new component in the Button menu in your sidebar 🍑



Now let's add the variations into our story so we can see all of the different shades of the Button that I have pre-built.

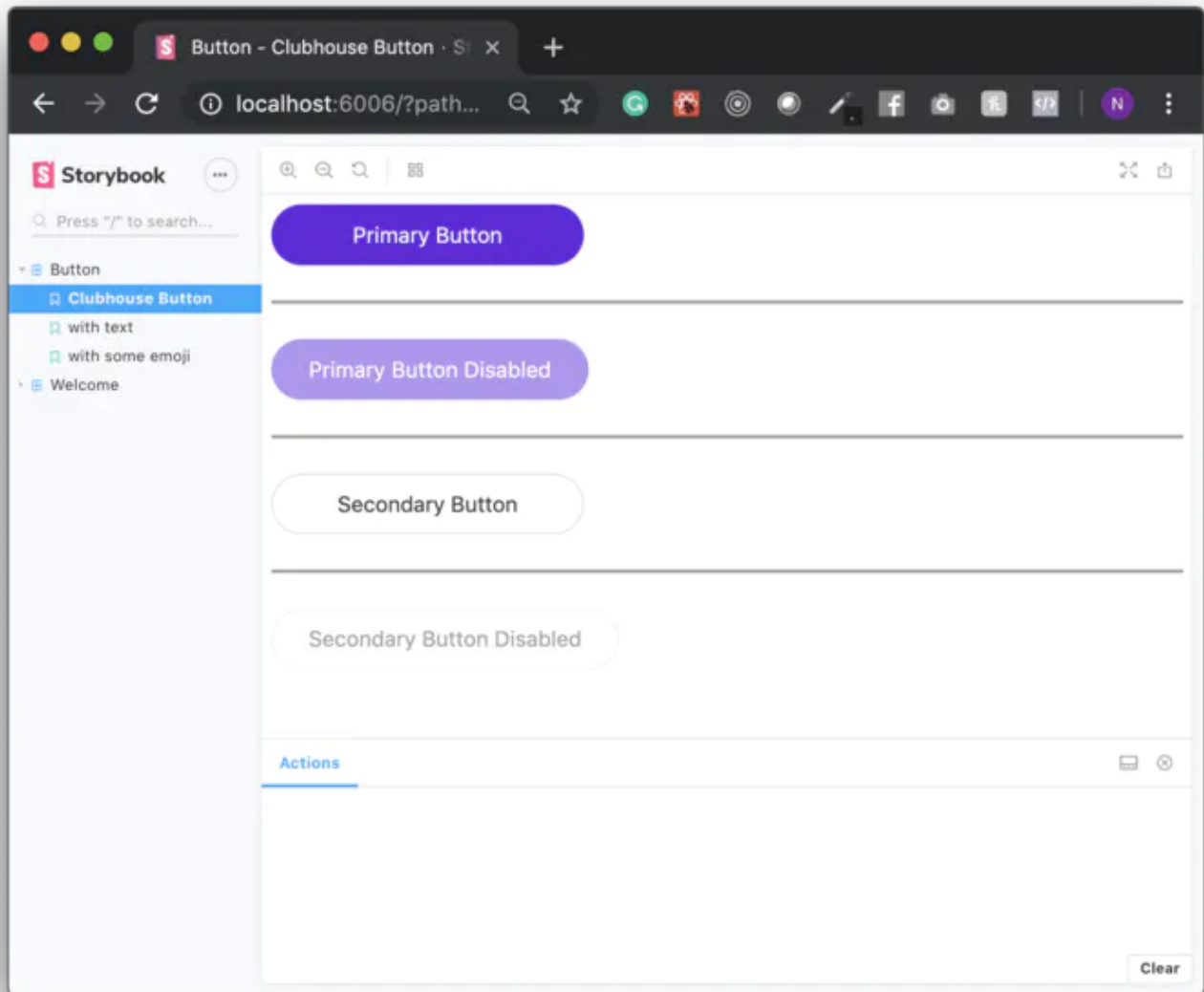
```
1  import React from 'react'
2  import { storiesOf } from '@storybook/react'
3  import { action } from '@storybook/addon-actions'
4  import Button from './'
5
6  const seperatorStyles = {
7    height: 2,
8    background: '#999',
9    margin: '30px 0'
10 }
11
12 storiesOf('Button', module)
13   .add('Clubhouse Button', () => (
14     <div>
15       <Button onClick={action('clicked')}>Primary Button</Button>
16       <div style={seperatorStyles} />
```

```
17     <Button disabled onClick={action('clicked')}>Primary Button Disabled</Button>
18     <div style={separatorStyles} />
19     <Button secondary onClick={action('clicked')}>Secondary Button</Button>
20     <div style={separatorStyles} />
21     <Button secondary disabled onClick={action('clicked')}>Secondary Button Disabled</Bu
22 </div>
23   )
24 }
```

Button.stories.js hosted with ❤ by GitHub

[view raw](#)

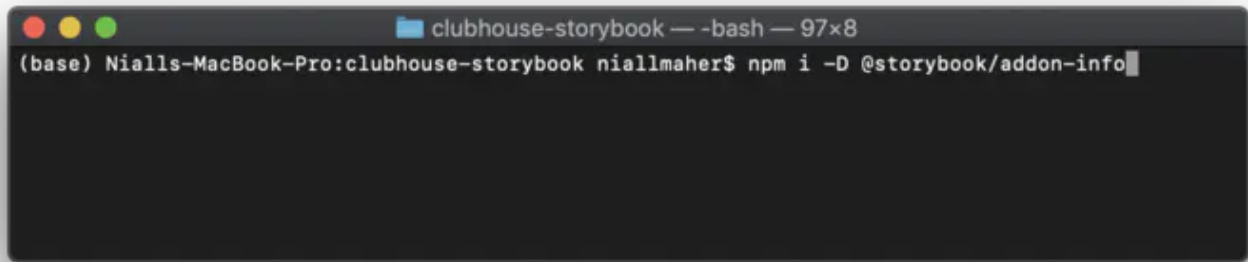
*(The divs are just there to create spacing between the components and make the story look a little prettier.)*



The real power begins when we throw in a few add-ons. You can find the list of all of the available add-ons [here](#). We already have the `action` addon out of the box but let's add a new one to show you how easy it is.

One addon I always use in development and to show you how to get started with plugins is [Info](#). Let's throw it in and you will see why!

Install the Info plugin with `npm i @storybook/addon-info`.



First in our Button story, import `withInfo` which is a named import:

```
import { withInfo } from "@storybook/addon-info"
```

Then we chain a decorator and pass it `withInfo`

```
storiesOf('Button', module)
  .addDecorator(withInfo) // make sure it is the first deco
  .add('Clubhouse Button', () => ( ...
```

We can now pass a second argument to the add function which will take an object with an `info` property. The `info` takes a string as a value. We can use a [Template String](#) to create some structure to the documentation we pass to the `info`. The `info` is read as markdown in the storybook so feel free to make this easy to read by adding appropriate styling and emojis.

Here's the complete code snippet for reference:

```
1 import React from 'react'
2 import { storiesOf } from '@storybook/react'
3 import { action } from '@storybook/addon-actions'
4 import { withInfo } from "@storybook/addon-info";
5 import Button from './'
```

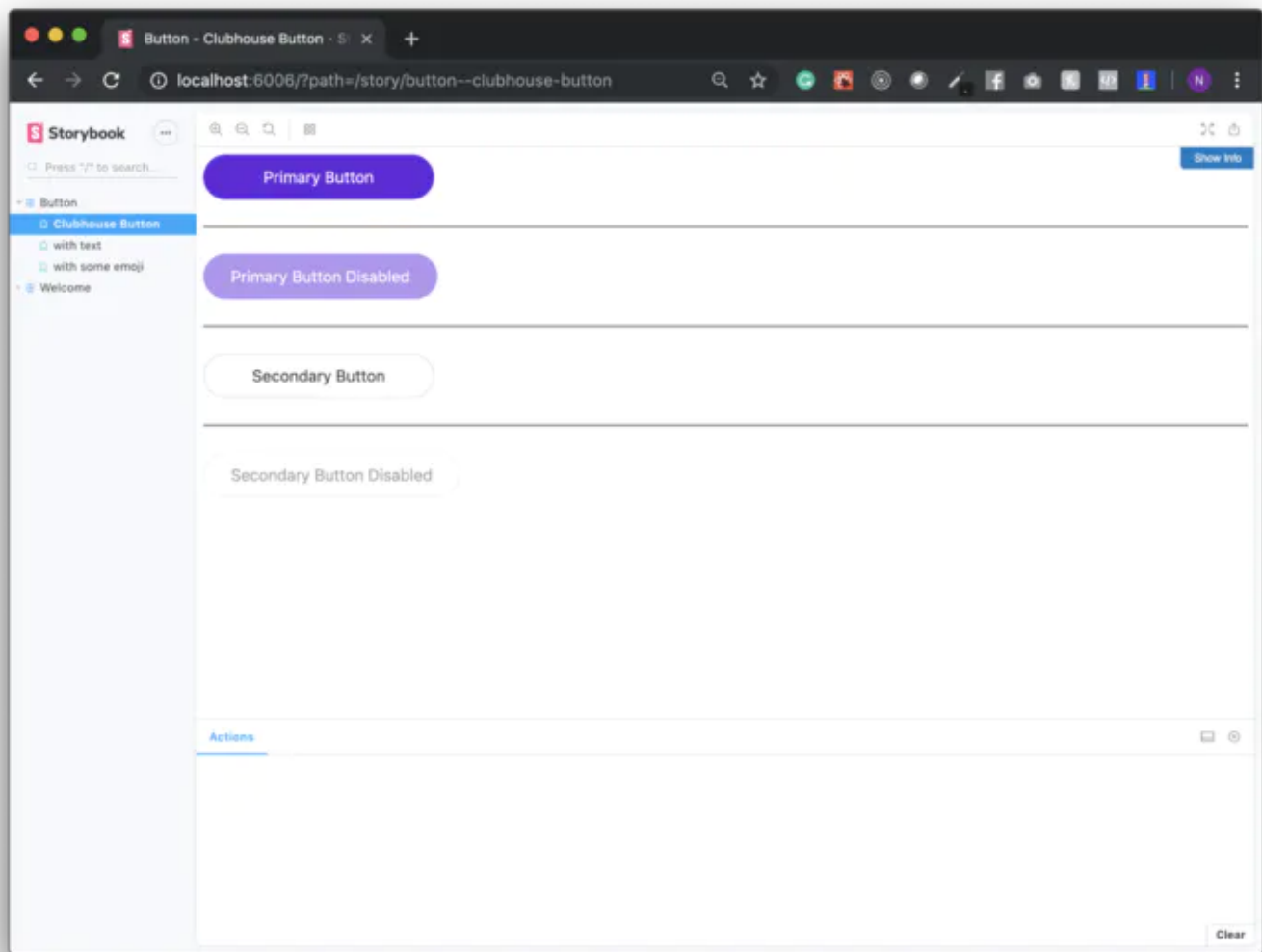
```
6
7  const seperatorStyles = {
8    height: 2,
9    background: '#999',
10   margin: '30px 0'
11 }
12
13 storiesOf('Button', module)
14   .addDecorator(withInfo)
15   .add('Clubhouse Button', () => (
16     <div>
17       <Button onClick={action('clicked')}>Primary Button</Button>
18       <div style={seperatorStyles} />
19       <Button disabled onClick={action('clicked')}>Primary Button Disabled</Button>
20       <div style={seperatorStyles} />
21       <Button secondary onClick={action('clicked')}>Secondary Button</Button>
22       <div style={seperatorStyles} />
23       <Button secondary disabled onClick={action('clicked')}>Secondary Button Disabled</Bu
24     </div>
25   ),
26   {info: `
27     📖 Clubhouse Button Guidelines 🙌
28
29     Should be used for all major call to actions.
30
31     The Clubhouse "secondary" style button should only be used when the background is
32     `
33   })
```

Button.stories.js hosted with ❤️ by GitHub

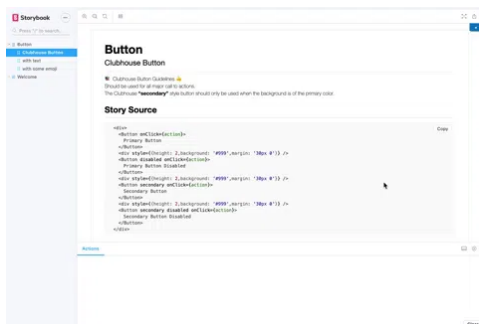
[view raw](#)

As a note, the `info` is a perfect place to add style guide notes so that developers know when to use these components. We will now see in our story preview a new button in the top right corner to “Show info”.





When you click this you will see we have a fairly comprehensive documentation set.



In the info section, you can copy the code snippets, see your documentation you passed with you `add-info` addon. The `prop-types` automatically generate a table showing if components are required and the type they expect. Another cool thing to note is in the propTypes table, you will see a “description” field. This is automatically generated when you structure your comments over your `prop-types` using a double star at the start of your comment such as:

```
Button.propTypes = {  
  /** Component takes a child node */  
  children: PropTypes.node.isRequired,
```

This, as you can see, takes minimal effort and allows us to take care of documentation as we develop (which is something we all dread writing). Adding usage notes from your designers, easily onboarding new hires and engineering productivity in general will be at an all-time high.

Find the completed code [here](#).

If you have made it this far, thanks for reading!

Share this Developer How-to



**1 Comment**   **Clubhouse****1 Login** ▾ **Recommend**    **Tweet**    **Share****Sort by Best** ▾

Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS **Stephen Ó Connor** • 7 months ago • edited

noop?

^ | ▾ • Reply • Share ▾

 **Subscribe**    **Add Disqus to your site** Add Disqus Add    **Disqus' Privacy Policy** Privacy Policy Privacy Policy

Join the over 25,000 developers, designers,  
and product managers already using  
Clubhouse 🚩

Get started—free forever

 Sign up with Google



## Product

[Features](#)[Pricing](#)[Enterprise](#)[Integrations](#)[Write beta](#)

## Company

[About Us](#)[Customers](#)[Careers](#)[Press](#)[Contact](#)[Brand Guidelines](#)[FAQ](#)

## Developers

[REST API Docs](#)[Webhook API Docs](#)[Community](#)[Open Source Projects](#)[Developer How-Tos](#)[Write for Clubhouse](#)

## Resources

[Help Center](#)[Blog](#)[Webinars](#)[Status](#)[Referral Program](#)[Release Notes](#)

## The Clubhouse Blog

[Collaborate faster at scale with Group @mentions and notifications](#)[Execute commands more efficiently with the Action Bar](#)

© 2020 Clubhouse Software Inc.

[Privacy Policy](#)

[Terms of Use](#)

[Security](#)

[Your Cookies](#)