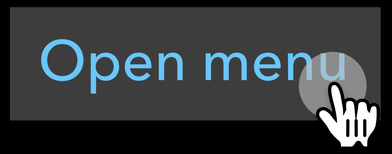
REINFORCING YOUR BLOCKS

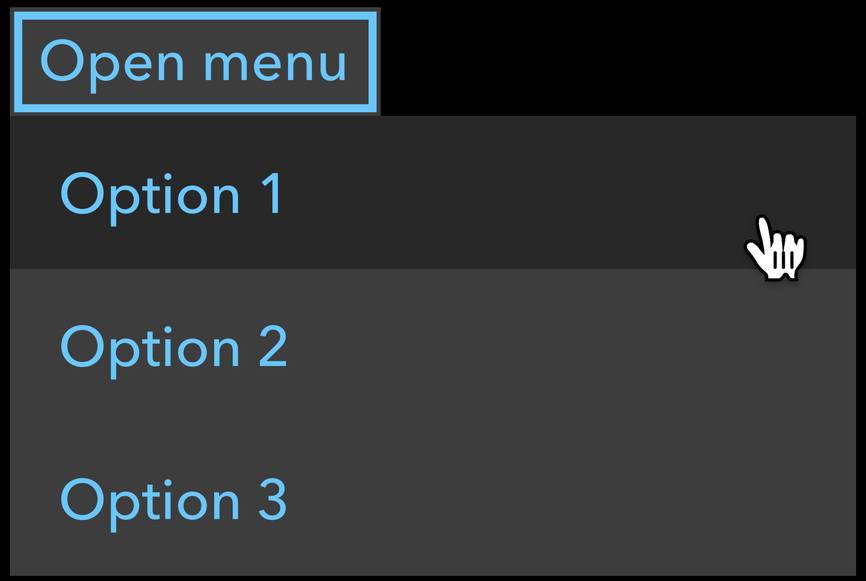
A practical guide to UI component testing with Jest and Storybook

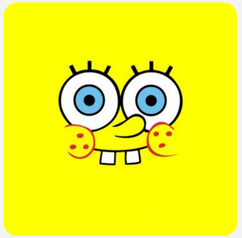
Hi. I'm Diego.

UI components

Open menu


Open menu


Open menu
Option 1
Option 2
Option 3

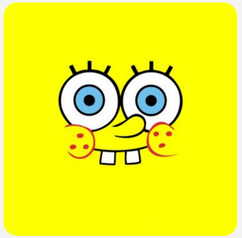


SpongeBob SquarePants

Fry cook – Krusty Krab

Online

Contact SpongeBob

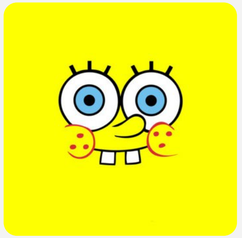


SpongeBob SquarePants

Fry cook – Krusty Krab

Online

Contact SpongeBob



SpongeBob SquarePants

Fry cook – Krusty Krab

Online

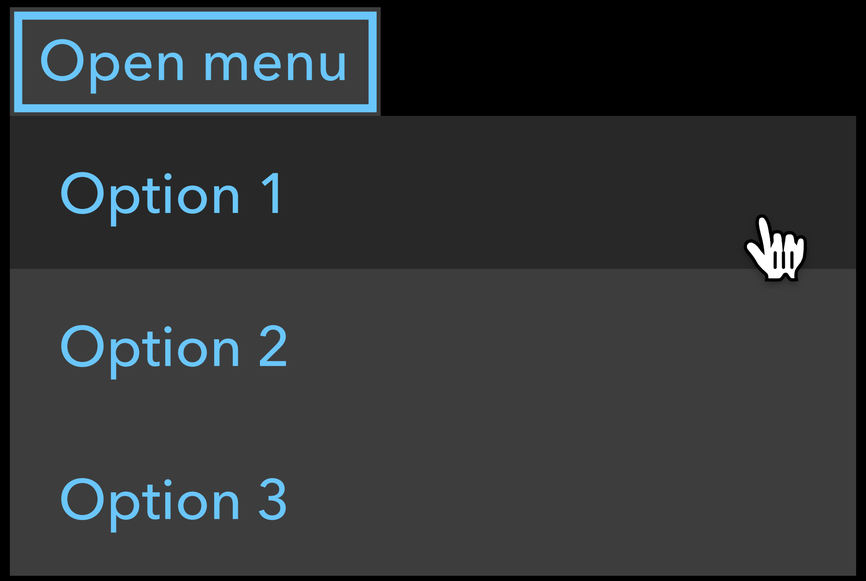
Contact SpongeBob

Start chat

Start voice call

Start video call

Does my UI work for my end-user?


Open menu
Option 1
Option 2
Option 3

Manual testing with Storybook

Automated testing with Jest
and DOM Testing Library

This talk is for all JS frameworks

React Vue Angular

Manual Testing with Storybook

package.json

```
{
  "devDependencies": {
    "@storybook/react": "^5.0.0",
    ...
  },
  "scripts": {
    "storybook": "start-storybook",
    ...
  },
  ...
}
```


.storybook
└── config.js
src
└── OverflowMenu
├── OverflowMenu.stories.js
├── OverflowMenu.test.js
└── index.js
└── setupTests.js

.storybook/config.js

```
import { configure } from '@storybook/react';

const loadStories = () => {
  const req = require.context(
    '../src', true, /\.stories\.js$/
  );
  req.keys().forEach(filename => req(filename));
}

configure(loadStories, module);
```

.storybook/config.js

```
import { configure } from '@storybook/react';

const loadStories = () => {
  const req = require.context(
    '../src', true, /\.stories\.js$/
  );
  req.keys().forEach(filename => req(filename));
}

configure(loadStories, module);
```

```
$ npm run storybook
```

```
Storybook 5.1.9 started
7.2 s for manager and 6.8 s for preview

Local:            http://localhost:9000/
On your network:  http://192.168.1.66:9000/
```

```
$ npm run storybook
```

```
Storybook 5.1.9 started
7.2 s for manager and 6.8 s for preview

Local:            http://localhost:9000/
On your network:  http://192.168.1.66:9000/
```

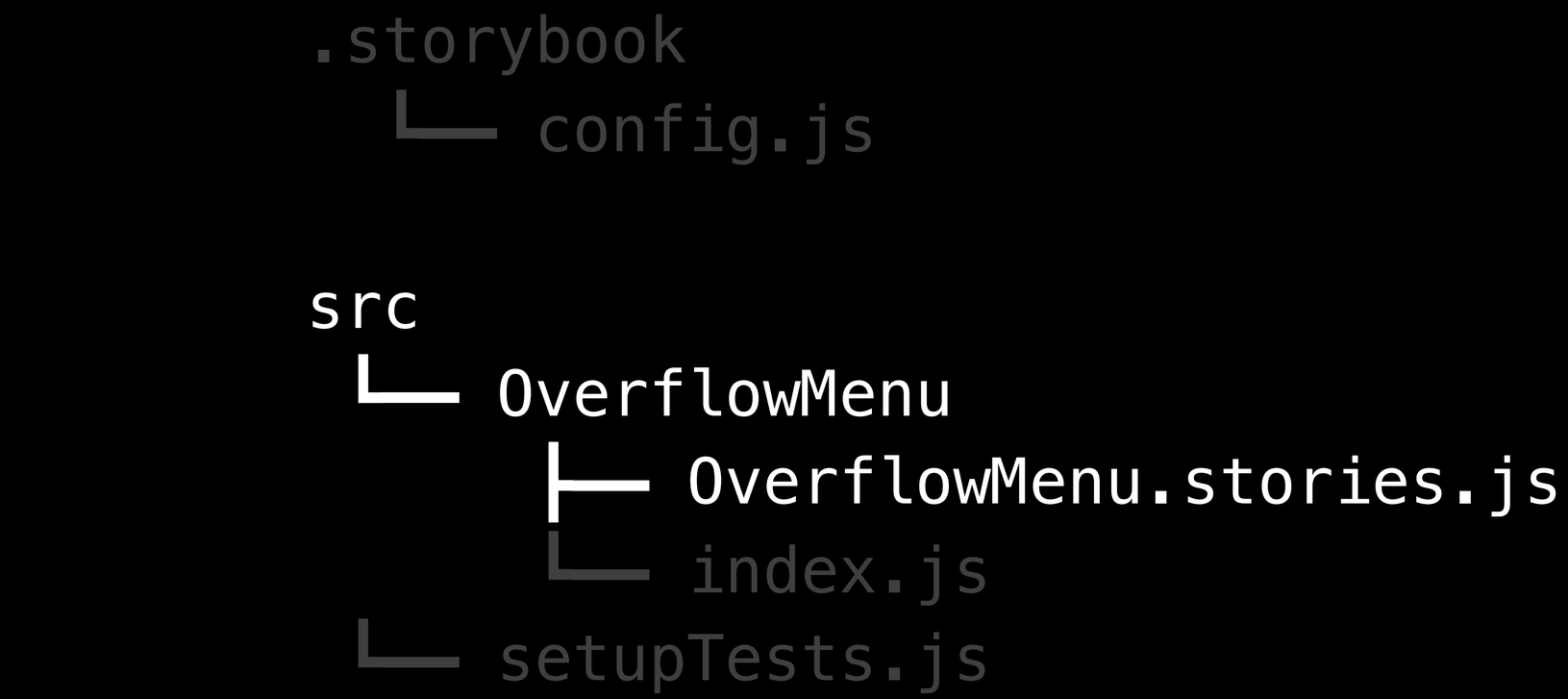
Storybook

No Preview

Sorry, but you either have no stories or none are selected somehow.

- Please check the Storybook config.
- Try reloading the page.

If the problem persists, check the browser console, or the terminal you've run Storybook from.


.storybook
└── config.js
src
└── OverflowMenu
├── OverflowMenu.stories.js
└── index.js
└── setupTests.js

src/OverflowMenu/OverflowMenu.stories.js

src/OverflowMenu/OverflowMenu.stories.js

```
import React from 'react';
import { storiesOf } from '@storybook/react';
import { OverflowMenu, MenuItem } from './OverflowMenu.js';
```

src/OverflowMenu/OverflowMenu.stories.js

```
import React from 'react';
import { storiesOf } from '@storybook/react';
import { OverflowMenu, MenuItem } from './OverflowMenu.js';

const stories = storiesOf('OverflowMenu', module);
```

src/OverflowMenu/OverflowMenu.stories.js

```
import React from 'react';
import { storiesOf } from '@storybook/react';
import { OverflowMenu, MenuItem } from './OverflowMenu.js';

const stories = storiesOf('OverflowMenu', module);

stories.add('default', () => (




));
```

src/OverflowMenu/OverflowMenu.stories.js

```
import React from 'react';
import { storiesOf } from '@storybook/react';
import { OverflowMenu, MenuItem } from './OverflowMenu.js';

const stories = storiesOf('OverflowMenu', module);

stories.add('default', () => (
  <OverflowMenu toggleText="Open menu">
    <MenuItem>Option 1</MenuItem>
    <MenuItem>Option 2</MenuItem>
    <MenuItem>Option 3</MenuItem>
  </OverflowMenu>
));
```

src/OverflowMenu/OverflowMenu.stories.js

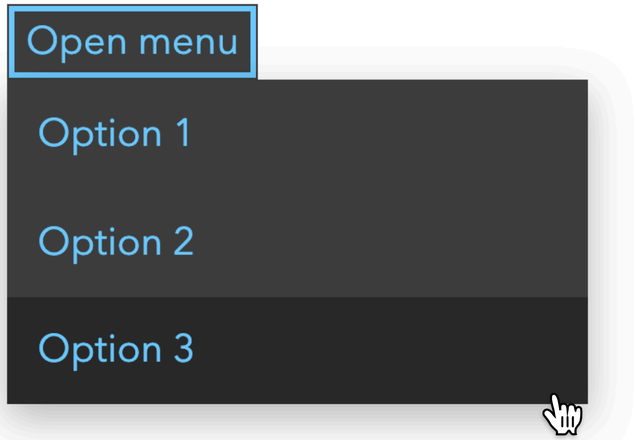
```
import React from 'react';
import { storiesOf } from '@storybook/react';
import { OverflowMenu, MenuItem } from './OverflowMenu.js';

const stories = storiesOf('OverflowMenu', module);

stories.add('default', () => (
  <OverflowMenu toggleText="Open menu">
    <MenuItem>Option 1</MenuItem>
    <MenuItem>Option 2</MenuItem>
    <MenuItem>Option 3</MenuItem>
  </OverflowMenu>
));
```

Storybook

Press "/" to search...

OverflowMenu

default



src/OverflowMenu/OverflowMenu.stories.js

```
stories.add('flipped', () => (
  <OverflowMenu
    toggleText="Open menu"
    flipped={true}
   >
    <MenuItem>Option 1</MenuItem>
    <MenuItem>Option 2</Item>
    <MenuItem>Option 3</MenuItem>
  </OverflowMenu>
));
```

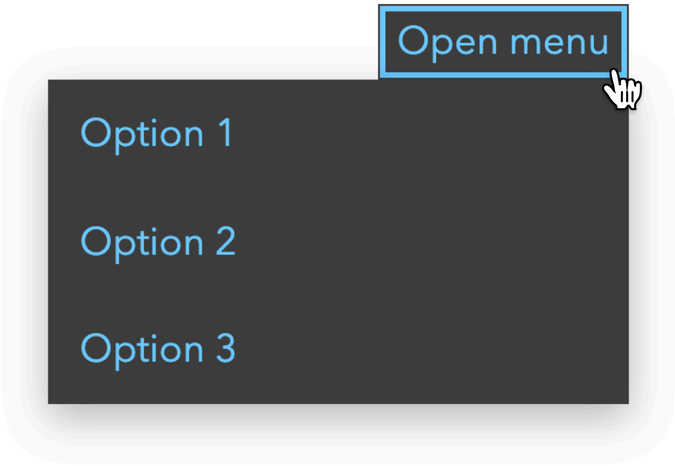
src/OverflowMenu/OverflowMenu.stories.js

```
stories.add('flipped', () => (
  <OverflowMenu
    toggleText="Open menu"
    flipped={true}
  >
    <MenuItem>Option 1</MenuItem>
    <MenuItem>Option 2</Item>
    <MenuItem>Option 3</MenuItem>
  </OverflowMenu>
));
```

Storybook

Press "/" to search...

- OverflowMenu
  - default
  - **flipped**



How is Storybook used for testing?

Render working components

Render working components

Test edge cases

Render working components

Test edge cases

Verify styles, animation, and interactions

Render working components

Test edge cases

Verify styles, animation, and interactions

Look for accessibility violations


Automated testing
with Jest and
DOM Testing Library


HI, HOW ARE YOU?

package.json

```
{
  "devDependencies": {
    "@testing-library/jest-dom": "^4.0.0",
    "@testing-library/react": "^8.0.0",
    "jest": "^24.0.0",
    ...
  },
  "scripts": {
    "test": "jest",
    ...
  },
  ...
}
```


.storybook
└── config.js
src
└── OverflowMenu
├── OverflowMenu.stories.js
└── index.js
└── setupTests.js

src/setupTests.js

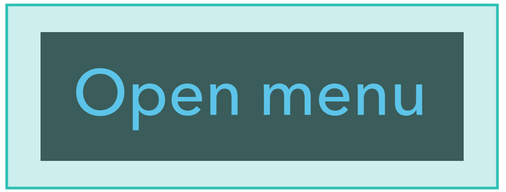
```
import '@testing-library/jest-dom/extend-expect';
import '@testing-library/react/cleanup-after-each';
```

Jest UI tests run in a simulated browser

Assert DOM element data

```
<OverflowMenu toggleText="Open menu">
  <MenuItem>Option 1</MenuItem>
</OverflowMenu>
```

```
<OverflowMenu toggleText="Open menu">
  <MenuItem>Option 1</MenuItem>
</OverflowMenu>
```




.storybook
└── config.js
src
└── OverflowMenu
├── OverflowMenu.stories.js
├── OverflowMenu.test.js
└── index.js
└── setupTests.js

src/OverflowMenu/OverflowMenu.test.js

src/OverflowMenu/OverflowMenu.test.js

```
import React from 'react';
import { render } from '@testing-library/react';
import OverflowMenu from './OverflowMenu.js';
```

src/OverflowMenu/OverflowMenu.test.js

```
import React from 'react';
import { render } from '@testing-library/react';
import OverflowMenu from './OverflowMenu.js';

test('has toggle text', () => {




});
```

src/OverflowMenu/OverflowMenu.test.js

```
import React from 'react';
import { render } from '@testing-library/react';
import OverflowMenu from './OverflowMenu.js';

test('has toggle text', () => {
  const { container } = render(
    <OverflowMenu toggleText="Open menu">
      <MenuItem>Option 1</MenuItem>
    </OverflowMenu>
  );

});
```

src/OverflowMenu/OverflowMenu.test.js

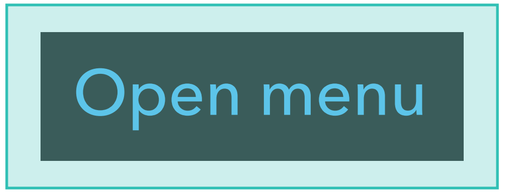
```
import React from 'react';
import { render } from '@testing-library/react';
import OverflowMenu from './OverflowMenu.js';

test('has toggle text', () => {
  const { container } = render(
    <OverflowMenu toggleText="Open menu">
      <MenuItem>Option 1</MenuItem>
    </OverflowMenu>
  );
  expect(container).toHaveTextContent('Open menu');
});
```

src/OverflowMenu/OverflowMenu.test.js

```
import React from 'react';
import { render } from '@testing-library/react';
import OverflowMenu from './OverflowMenu.js';

test('has toggle text', () => {
  const { container } = render(
    <OverflowMenu toggleText="Open menu">
      <MenuItem>Option 1</MenuItem>
    </OverflowMenu>
  );
  expect(container).toHaveTextContent('Open menu');
});
```

```
$ npm run test

 PASS  src/OverflowMenu/OverflowMenu.test.js
   ✓ has toggle text (20ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        20ms, estimated 1s
```

```
<OverflowMenu toggleText="Open menu">
  <MenuItem>Option 1</MenuItem>
</OverflowMenu>
```



```
<OverflowMenu toggleText="Open menu">
  <MenuItem>Option 1</MenuItem>
</OverflowMenu>
```

src/OverflowMenu/OverflowMenu.test.js

```
test('toggle button has toggle text', () => {
  const { getByTestId } = render(
    <OverflowMenu toggleText="Open menu">
      <MenuItem>Option 1</MenuItem>
    </OverflowMenu>
  );

  const toggleButton = getByTestId('overflow-toggle');

  expect(toggleButton).toHaveTextContent('Open menu');
});
```

./OverflowMenu.test.js

```
test('toggle button has toggle text', () => {
  const { getByTestId } = render(
    <OverflowMenu toggleText="Open menu">
      <MenuItem>Option 1</MenuItem>
    </OverflowMenu>
  );

  const toggleButton = getByTestId('overflow-toggle');

  expect(toggleButton).toHaveTextContent('Open menu');
});
```

./OverflowMenu.test.js

```
test('toggle button has toggle text', () => {
  const { getByTestId } = render(
    <OverflowMenu toggleText="Open menu">
      <MenuItem>Option 1</MenuItem>
    </OverflowMenu>
  );

  const toggleButton = getByTestId('overflow-toggle');

  expect(toggleButton).toHaveTextContent('Open menu');
});
```

```
container.querySelector('[data-testid="overflow-toggle"]');
```

./OverflowMenu.test.js

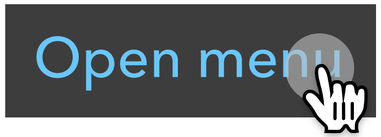
```
test('toggle button has toggle text', () => {
  const { getByTestId } = render(
    <OverflowMenu toggleText="Open menu">
      <MenuItem>Option 1</MenuItem>
    </OverflowMenu>
  );

  const toggleButton = getByTestId('overflow-toggle');

  expect(toggleButton).toHaveTextContent('Open menu');
});
```

```
$ npm run test

 PASS  src/OverflowMenu/OverflowMenu.test.js
   ✓ toggle button has toggle text (20ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        20ms, estimated 1s
```

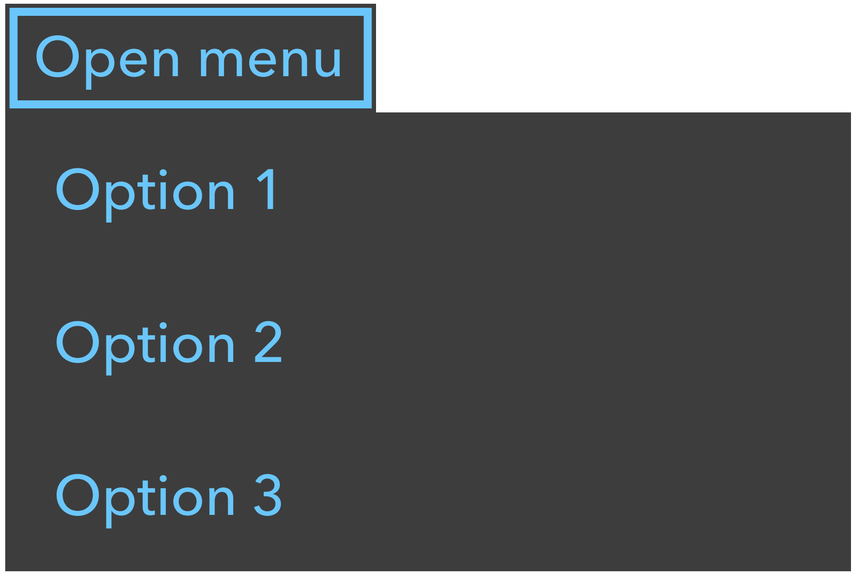
```
expect(element).toHaveClass('my-class');

expect(element).toBeVisible();

expect(element).toBeInDocument();

expect(element).toHaveStyle('color: #BEEBEE');
```

```
expect(element).not.toHaveClass('my-class');

expect(element).not.toBeVisible();

expect(element).not.toBeInDocument();

expect(element).not.toHaveStyle('color: #BEEBEE');
```

```
<OverflowMenu toggleText="Open menu">
  <MenuItem>Option 1</MenuItem>
  <MenuItem>Option 2</MenuItem>
  <MenuItem>Option 3</MenuItem>
</OverflowMenu>
```



```
<OverflowMenu toggleText="Open menu">
  <MenuItem>Option 1</MenuItem>
  <MenuItem>Option 2</MenuItem>
  <MenuItem>Option 3</MenuItem>
</OverflowMenu>
```



src/OverflowMenu/OverflowMenu.test.js

```
import { render } from '@testing-library/react';

test('shows items', () => {
  const { getByText } = render(
    <OverflowMenu toggleText="Open menu">
      <MenuItem>Option 1</MenuItem>
    </OverflowMenu>
  );

 expect(getByText('Option 1')).toBeVisible();
});
```

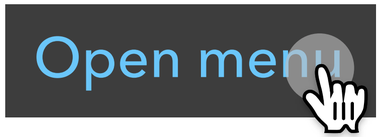
src/OverflowMenu/OverflowMenu.test.js

```
import { render } from '@testing-library/react';

test('shows items', () => {
  const { getByText } = render(
    <OverflowMenu toggleText="Open menu">
      <MenuItem>Option 1</MenuItem>
    </OverflowMenu>
  );

 expect(getByText('Option 1')).toBeVisible();
});
```

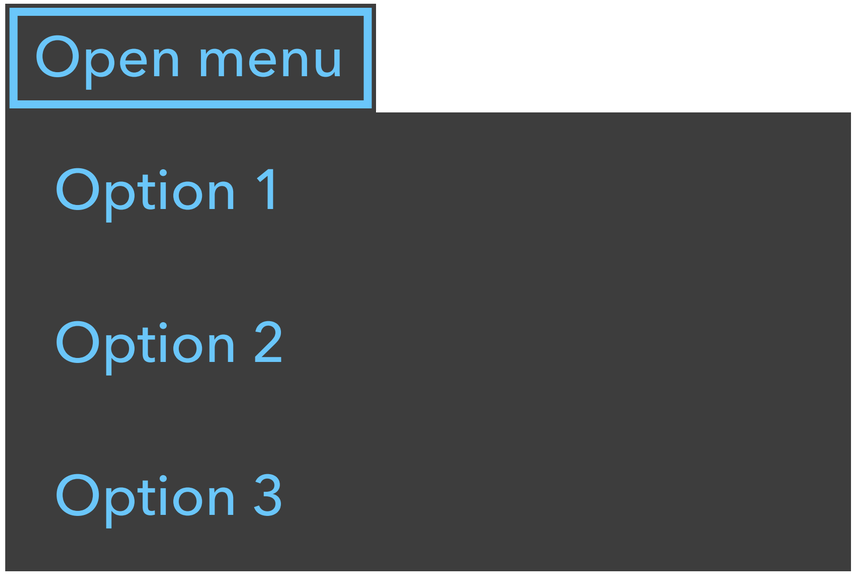
```
 FAIL  src/OverflowMenu/OverflowMenu.test.js
  ● shows items

    Unable to find an element with the text: Option 1. This could be because the text is broken up by multiple elements. In this case, you can provide a function for your text matcher to make your matcher more flexible.

      <body>
        <div>
          <div
            class="sc-bdVaJa huqire"
            data-testid="overflow-menu"
          >
```

```
 FAIL  src/OverflowMenu/OverflowMenu.test.js
  ● shows items

    Unable to find an element with the text: Option 1.
This could be because the text is broken up by multiple
elements. In this case, you can provide a function for
your text matcher to make your matcher more flexible.

      <body>
        <div>
          <div
            class="sc-bdVaJa huqire"
            data-testid="overflow-menu"
          >
```

```
<OverflowMenu toggleText="Open menu">
  <MenuItem>Option 1</MenuItem>
  <MenuItem>Option 2</MenuItem>
  <MenuItem>Option 3</MenuItem>
</OverflowMenu>
```



```
<OverflowMenu toggleText="Open menu">
  <MenuItem>Option 1</MenuItem>
  <MenuItem>Option 2</MenuItem>
  <MenuItem>Option 3</MenuItem>
</OverflowMenu>
```



src/OverflowMenu/OverflowMenu.test.js

```
import { render } from '@testing-library/react';

test('shows items', () => {
  const { getByText } = render(
    <OverflowMenu toggleText="Open menu">
      <MenuItem>Option 1</MenuItem>
    </OverflowMenu>
  );

 expect(getByText('Option 1')).toBeVisible();
});
```

src/OverflowMenu/OverflowMenu.test.js

```
import { render, fireEvent } from '@testing-library/react';

test('shows items when user clicks on toggle', () => {
  const { getByText } = render(
    <OverflowMenu toggleText="Open menu">
      <MenuItem>Option 1</MenuItem>
    </OverflowMenu>
  );

  fireEvent.click(getByText('Open menu'));
  expect(getByText('Option 1')).toBeVisible();
});
```

src/OverflowMenu/OverflowMenu.test.js

```
import { render, fireEvent } from '@testing-library/react';

test('shows items when user clicks on toggle', () => {
  const { getByText } = render(
    <OverflowMenu toggleText="Open menu">
      <MenuItem>Option 1</MenuItem>
    </OverflowMenu>
  );

  fireEvent.click(getByText('Open menu'));
  expect(getByText('Option 1')).toBeVisible();
});
```

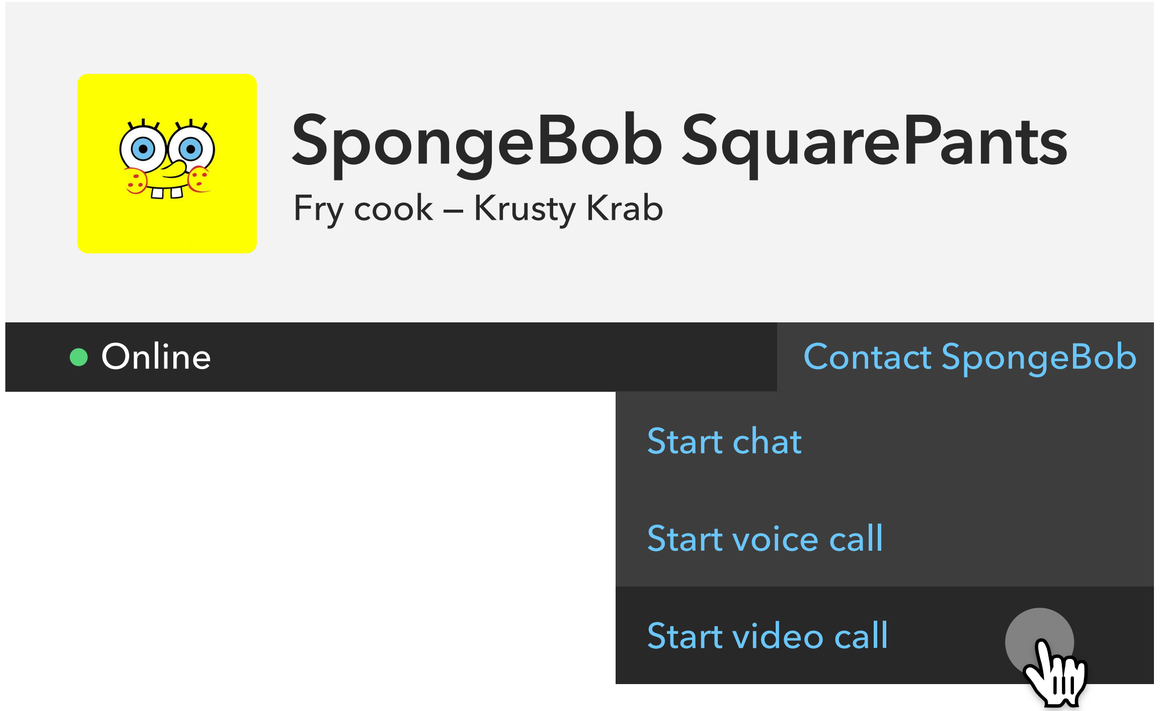
src/OverflowMenu/OverflowMenu.test.js

```
import { render, fireEvent } from '@testing-library/react';

test('shows items when user clicks on toggle', () => {
  const { getByText } = render(
    <OverflowMenu toggleText="Open menu">
      <MenuItem>Option 1</MenuItem>
    </OverflowMenu>
  );

  fireEvent.click(getByText('Open menu'));
  expect(getByText('Option 1')).toBeVisible();
});
```

```
$ npm run test

 PASS  src/OverflowMenu/OverflowMenu.test.js
   ✓ toggle button has toggle text (20ms)
   ✓ shows items when user clicks on toggle (22ms)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        42ms, estimated 1s
```

```
fireEvent.focus(element);

fireEvent.mouseEnter(element);

fireEvent.change(element, { target: { value: 'hello' }});

fireEvent.keyDown(element, { key: 'Enter' });
```

```
const startVideo = id => {
  api.startVideo(id);
};

<ContactCard
  {...contactInfo}
  onVideoCall={startVideo}
/>
```



src/ContactCard/ContactCard.test.js

```
test('initializes video call', () => {
  const onVideoMock = jest.fn();
  const { getByText } = render(
     <ContactCard onVideoCall={onVideoMock} {...contact} />
  );

  fireEvent.click(getByText('Contact'));
  fireEvent.click(getByText('Start video call'));

  expect(onVideoMock).toHaveBeenCalledTimes(1);
});
```

src/ContactCard/ContactCard.test.js

```
test('initializes video call', () => {
  const onVideoMock = jest.fn();
  const { getByText } = render(
    <ContactCard onVideoCall={onVideoMock} {...contact} />
  );

  fireEvent.click(getByText('Contact'));
  fireEvent.click(getByText('Start video call'));

  expect(onVideoMock).toHaveBeenCalledTimes(1);
});
```

src/ContactCard/ContactCard.test.js

```
test('initializes video call', () => {
  const onVideoMock = jest.fn();
  const { getByText } = render(
    <ContactCard onVideoCall={onVideoMock} {...contact} />
  );

  fireEvent.click(getByText('Contact'));
  fireEvent.click(getByText('Start video call'));

  expect(onVideoMock).toHaveBeenCalledTimes(1);
});
```

src/ContactCard/ContactCard.test.js

```
test('initializes video call', () => {
  const onVideoMock = jest.fn();
  const { getByText } = render(
    <ContactCard onVideoCall={onVideoMock} {...contact} />
  );

  fireEvent.click(getByText('Contact'));
  fireEvent.click(getByText('Start video call'));

  expect(onVideoMock).toHaveBeenCalledTimes(1);
});
```

src/ContactCard/ContactCard.test.js

```
test('initializes video call', () => {
  const onVideoMock = jest.fn();
  const { getByText } = render(
     <ContactCard onVideoCall={onVideoMock} {...contact} />
  );

  fireEvent.click(getByText('Contact'));
  fireEvent.click(getByText('Start video call'));

  expect(onVideoMock).toHaveBeenCalledTimes(1);
});
```

Why is testing with Jest beneficial?

Automate rendering tests

Automate rendering tests

Simulate user behavior (`fireEvent`)

Automate rendering tests

Simulate user behavior (`fireEvent`)

Verify callback prop calls with mocks (`onVideoCall`)

Automate rendering tests

Simulate user behavior (`fireEvent`)

Verify callback prop calls with mocks (`onVideoCall`)

100s of tests in seconds

Conclusion

Storybook

✘ Automated

✔ Rendering

✔ Styles & animation

✔ Accessibility

✔ Interactions

✔ Actions

Jest

✔ Automated

✔ Rendering

✘ Styles & animations

✘ Accessibility

✔ Interactions

✔ Actions

Storybook

✘ Automated

✔ Rendering

✔ Styles & animation

✔ Accessibility

✔ Interactions

✔ Actions

Jest

✔ Automated

✔ Rendering

✘ Styles & animations

✘ Accessibility

✔ Interactions

✔ Actions

Check the docs

storybook.js.org

jestjs.io

testing-library.com

Thank you!