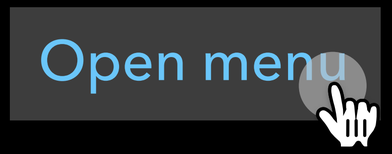
REINFORCING YOUR BLOCKS

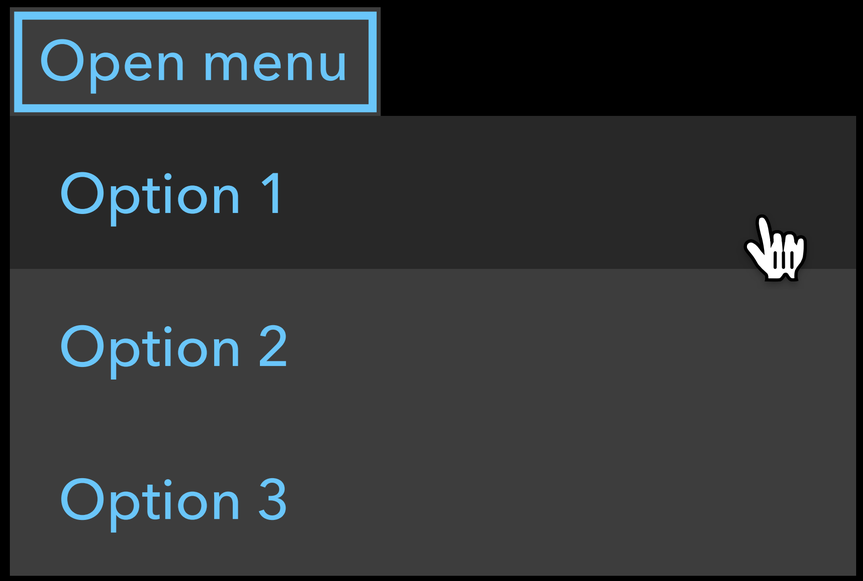
A practical guide to UI component testing with Jest and Storybook

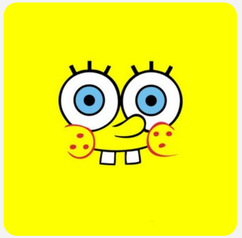
Hi. I'm Diego.

UI components

Open menu


Open menu


Open menu
Option 1
Option 2
Option 3



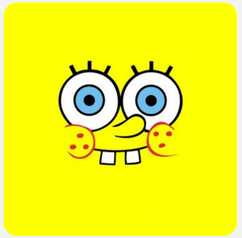
SpongeBob SquarePants

Fry cook – Krusty Krab

● Online

Contact SpongeBob


SpongeBob SquarePants
Fry cook – Krusty Krab
Online
Contact SpongeBob



SpongeBob SquarePants

Fry cook – Krusty Krab

● Online

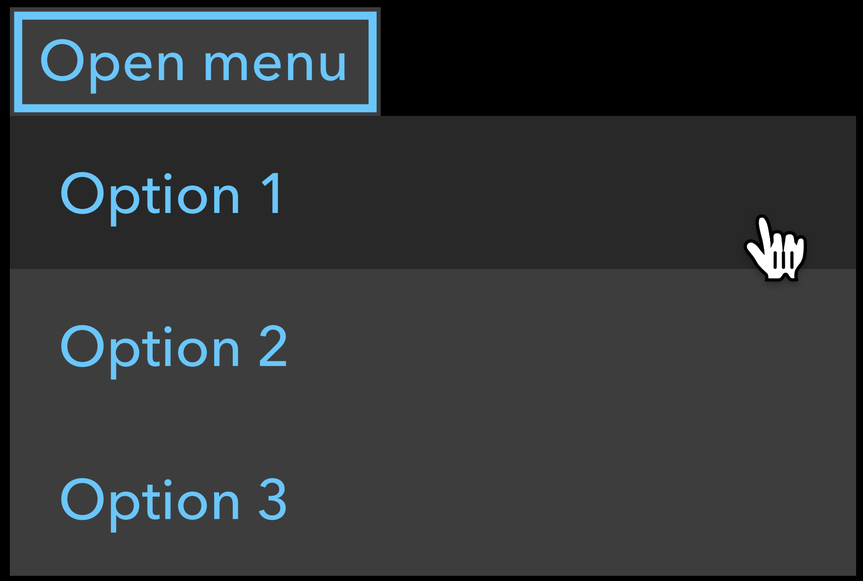
Contact SpongeBob

Start chat

Start voice call

Start video call

Does the UI work for the end-user?


Open menu
Option 1
Option 2
Option 3

Manual testing with Storybook

Automated unit 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": "^5.2.5",
    ...
  },
  "scripts": {
    "storybook": "start-storybook",
    ...
  },
  ...
}
```

```
$ 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

Canvas Docs

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 { OverflowMenu, MenuItem } from './OverflowMenu.js';
```

src/OverflowMenu/OverflowMenu.stories.js

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

export default { title: 'OverflowMenu' };
```

src/OverflowMenu/OverflowMenu.stories.js

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

export default { title: 'OverflowMenu' };

export const base = () => (




);
```

src/OverflowMenu/OverflowMenu.stories.js

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

export default { title: 'OverflowMenu' };

export const base = () => (
  <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 { OverflowMenu, MenuItem } from './OverflowMenu.js';

export default { title: 'OverflowMenu' };

export const base = () => (
  <OverflowMenu toggleText="Open menu">
    <MenuItem>Option 1</MenuItem>
    <MenuItem>Option 2</MenuItem>
    <MenuItem>Option 3</MenuItem>
  </OverflowMenu>
);
```

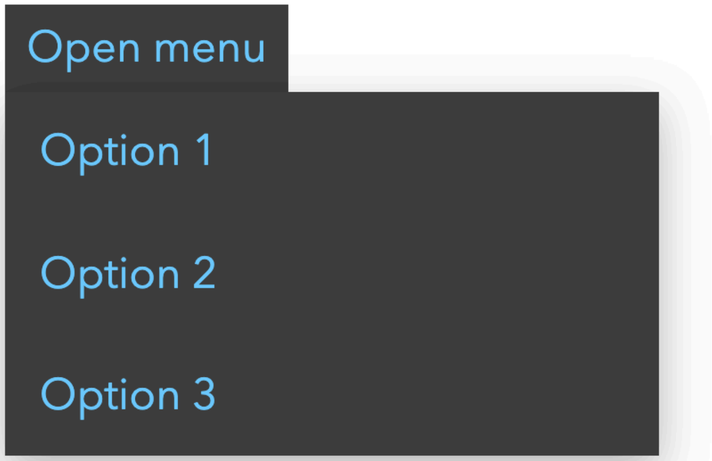
Storybook

Press "/" to search...

OverflowMenu

Base

Canvas Docs



src/OverflowMenu/OverflowMenu.stories.js

```
expect const 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

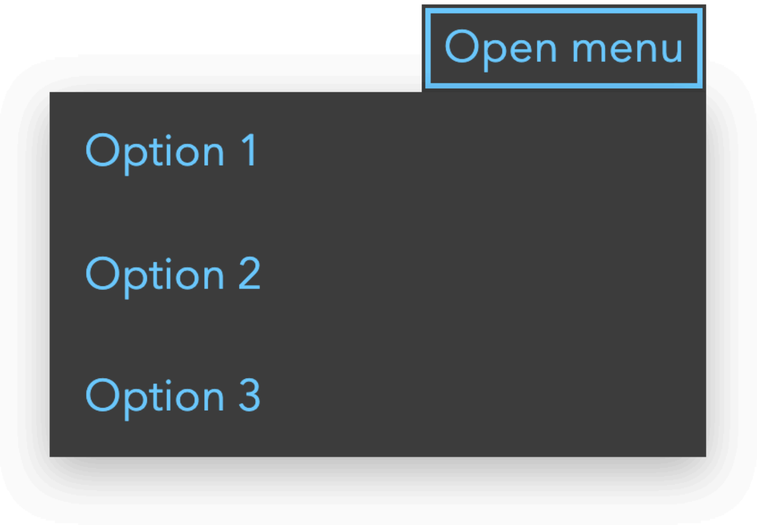
```
expect const flipped = () => (
  <OverflowMenu
    toggleText="Open menu"
    flipped={true}
  >
    <MenuItem>Option 1</MenuItem>
    <MenuItem>Option 2</Item>
    <MenuItem>Option 3</MenuItem>
  </OverflowMenu>
);
```

Storybook

Press "/" to search...

- OverflowMenu
  - Base
  - **Flipped**

Canvas | Docs

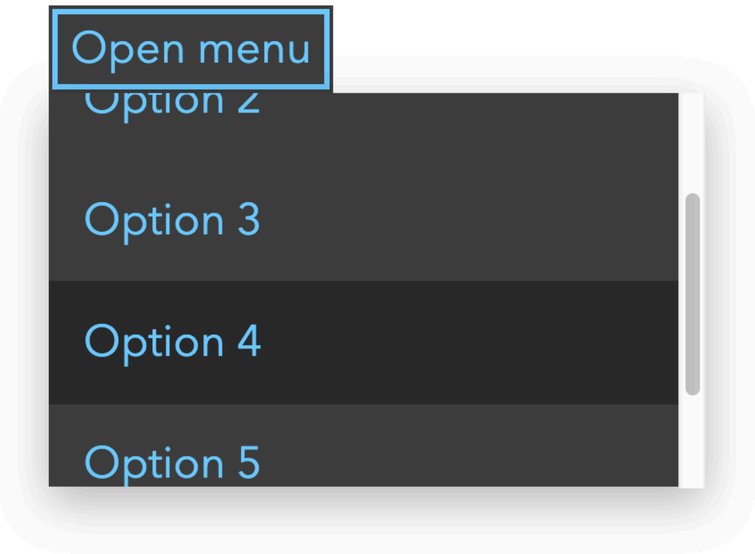


Storybook

Press "/" to search...

- OverflowMenu
  - Base
  - Flipped
  - **With Lots Of Items**
  - With Long Toggle Text

Canvas | Docs

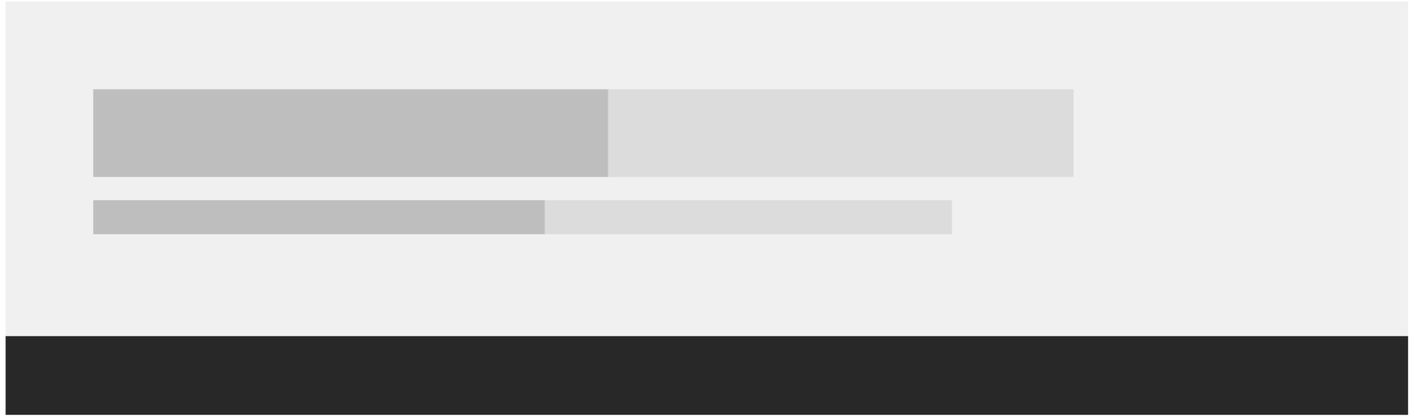


Storybook

Press "/" to search...

- ContactCard
  - Base
  - Online
  - With Phone Number
  - With Email
  - Online With All Contact Info
  - **Loading**
- OverflowMenu

Canvas | Docs



Storybook addons

Why is testing with Storybook beneficial?

Render working isolated UI

Render working isolated UI

Preview UI in hard-to-replicate states

Render working isolated UI

Preview UI in hard-to-replicate states

Verify styles, animation, and interactions

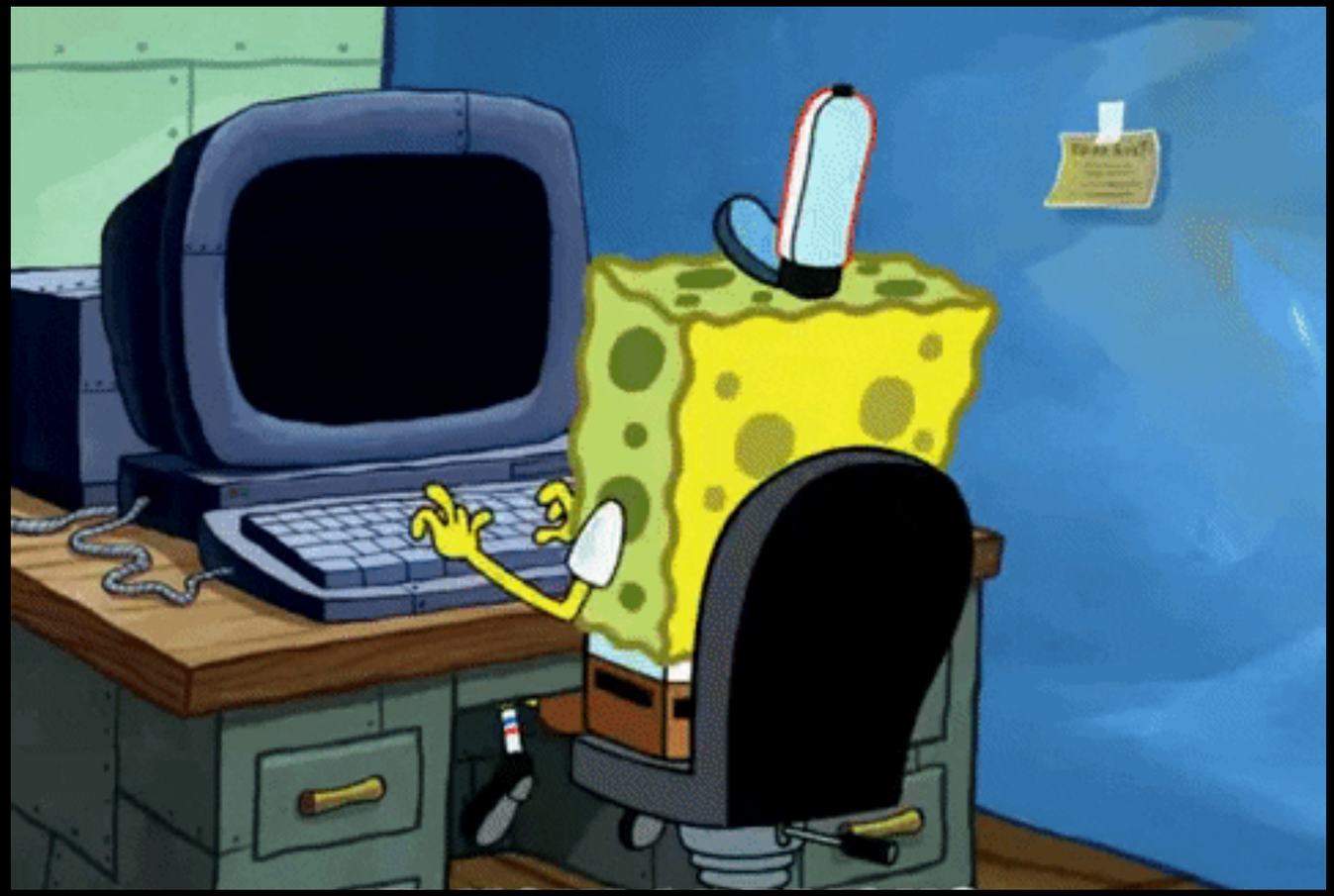
Render working isolated UI

Preview UI in hard-to-replicate states

Verify styles, animation, and interactions

Publish as a static site


Automated testing
with Jest and
DOM Testing Library



package.json

```
{
  "devDependencies": {
    "@testing-library/jest-dom": "^4.1.2",
    "@testing-library/react": "^9.3.0",
    "jest": "^24.9.0",
    ...
  },
  "scripts": {
    "test": "jest",
    ...
  },
  ...
}
```

Jest UI tests run in a simulated browser

```
<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, MenuItem } from './OverflowMenu.js';
```

src/OverflowMenu/OverflowMenu.test.js

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

test('options are not visible by default', () => {




});
```

src/OverflowMenu/OverflowMenu.test.js

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

test('options are not visible by default', () => {
  const { queryByText } = 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, MenuItem } from './OverflowMenu.js';

test('options are not visible by default', () => {
  const { queryByText } = render(
    <OverflowMenu toggleText="Open menu">
      <MenuItem>Option 1</MenuItem>
    </OverflowMenu>
  );
  expect(                              )
});
```

src/OverflowMenu/OverflowMenu.test.js

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

test('options are not visible by default', () => {
  const { queryByText } = render(
    <OverflowMenu toggleText="Open menu">
      <MenuItem>Option 1</MenuItem>
    </OverflowMenu>
  );
  expect(queryByText('Option 1'))
});
```

src/OverflowMenu/OverflowMenu.test.js

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

test('options are not visible by default', () => {
  const { queryByText } = render(
    <OverflowMenu toggleText="Open menu">
      <MenuItem>Option 1</MenuItem>
    </OverflowMenu>
  );
  expect(queryByText('Option 1')).not.toBeInTheDocument();
});
```

src/OverflowMenu/OverflowMenu.test.js

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

test('options are not visible by default', () => {
  const { queryByText } = render(
    <OverflowMenu toggleText="Open menu">
      <MenuItem>Option 1</MenuItem>
    </OverflowMenu>
  );
  expect(queryByText('Option 1')).not.toBeInTheDocument();
});
```

```
$ npm run test

 PASS  src/OverflowMenu/OverflowMenu.test.js
   ✓ options are not visible by default (20ms)

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

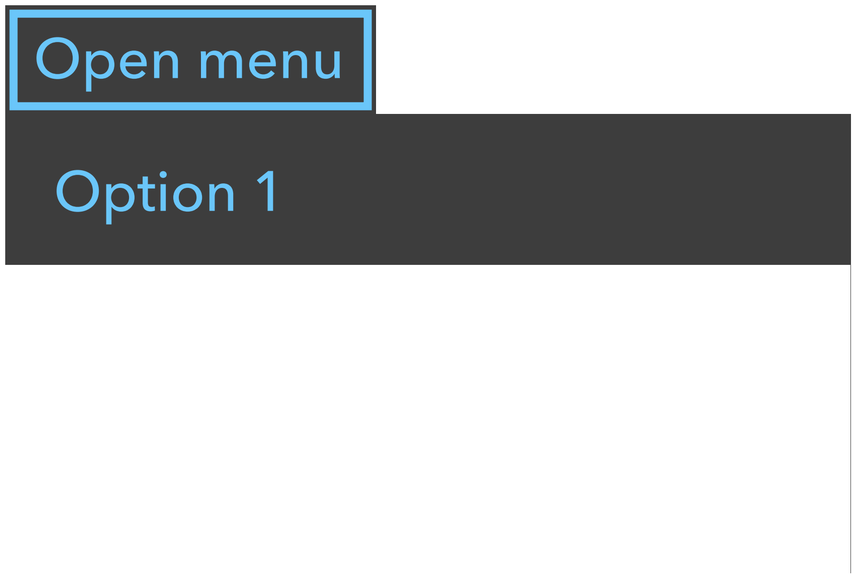
Use UI labels to find DOM elements

src/OverflowMenu/OverflowMenu.test.js

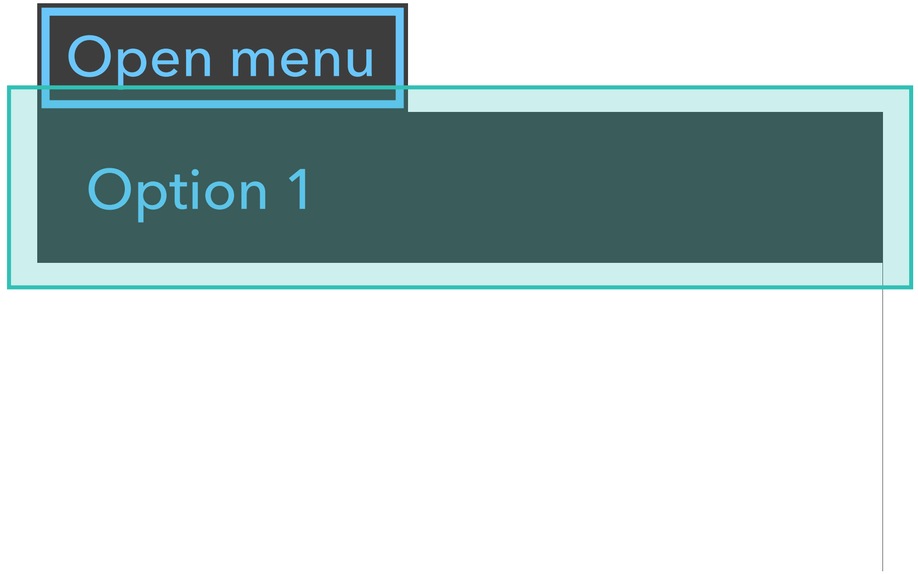
```
import React from 'react';
import { render } from '@testing-library/react';
import { OverflowMenu, MenuItem } from './OverflowMenu.js';

test('options are not visible by default', () => {
  const { queryByText } = render(
    <OverflowMenu toggleText="Open menu">
      <MenuItem>Option 1</MenuItem>
    </OverflowMenu>
  );
  expect(queryByText('Option 1')).not.toBeInTheDocument();
});
```

Test state changes

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



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



src/OverflowMenu/OverflowMenu.test.js

```
test('options are visible when toggle is clicked', () => {
  const { getByText } = render(
    <OverflowMenu toggleText="Open menu">
      <MenuItem>Option 1</MenuItem>
    </OverflowMenu>
  );
  expect(getByText('Option 1')).toBeVisible();
});
```

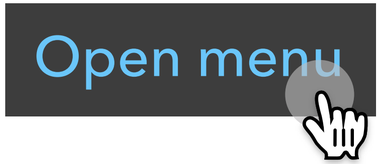
src/OverflowMenu/OverflowMenu.test.js

```
test('options are visible when toggle is clicked', () => {
  const { getByText } = render(
    <OverflowMenu toggleText="Open menu">
      <MenuItem>Option 1</MenuItem>
    </OverflowMenu>
  );
  expect(getByText('Option 1')).toBeVisible();
});
```

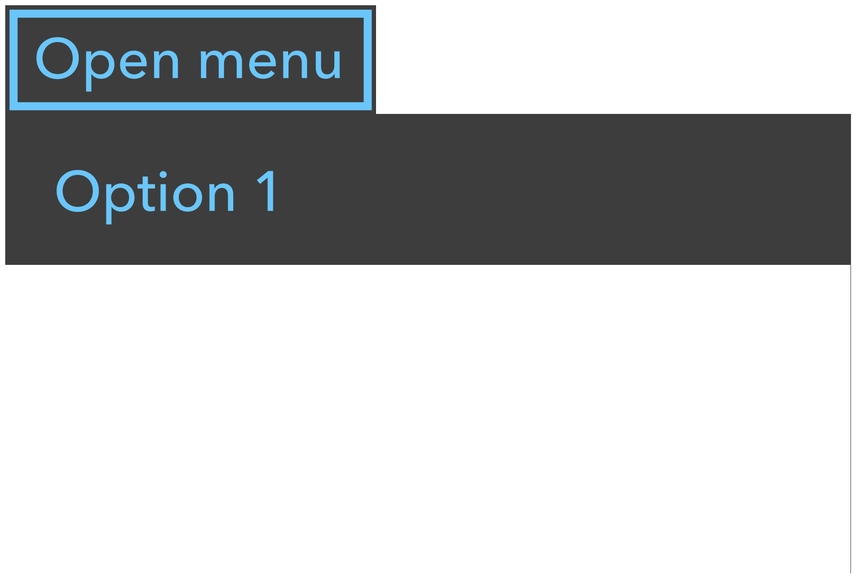
```
 FAIL  src/OverflowMenu/OverflowMenu.test.js
  ● options are visible when toggle is clicked

    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"
          >
            <button
```

```
 FAIL  src/OverflowMenu/OverflowMenu.test.js
  ● options are visible when toggle is clicked

    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"
          >
            <button
```

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



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



src/OverflowMenu/OverflowMenu.test.js

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

test('options are visible when toggle is clicked', () => {
  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('options are visible when toggle is clicked', () => {
  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('options are visible when toggle is clicked', () => {
  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('options are visible when toggle is clicked', () => {
  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('options are visible when toggle is clicked', () => {
  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
   ✓ options are not visible by default (20ms)
   ✓ options are visible when toggle is clicked (22ms)

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

Make assertions as specific as possible

src/OverflowMenu/OverflowMenu.test.js

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

test('options are visible when toggle is clicked', () => {
  const { getByText } = render(
    <OverflowMenu toggleText="Open menu">
      <MenuItem>Option 1</MenuItem>
    </OverflowMenu>
  );
  fireEvent.click(getByText('Open menu'));
  expect(getByText('Option 1')).toBeInTheDocument();
});
```

src/OverflowMenu/OverflowMenu.test.js

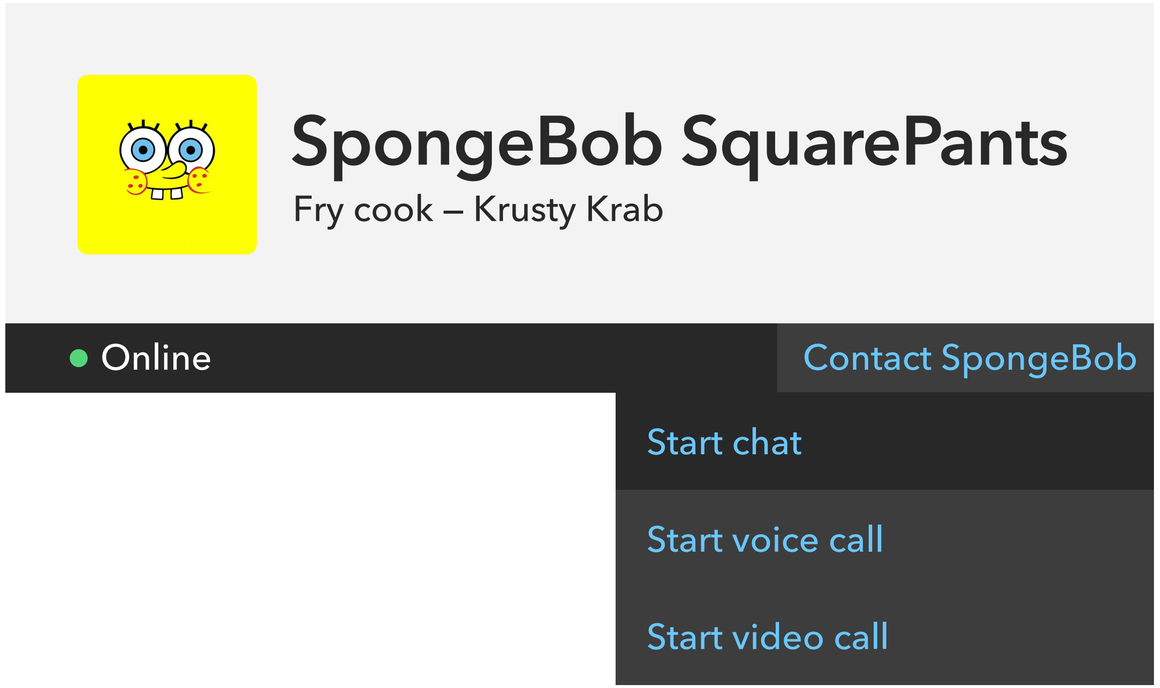
```
import { render, fireEvent } from '@testing-library/react';

test('options are visible when toggle is clicked', () => {
  const { getByText } = render(
    <OverflowMenu toggleText="Open menu">
      <MenuItem>Option 1</MenuItem>
    </OverflowMenu>
  );
  fireEvent.click(getByText('Open menu'));
  expect(getByText('Option 1')).toBeInTheDocument();
});
```

src/OverflowMenu/OverflowMenu.test.js

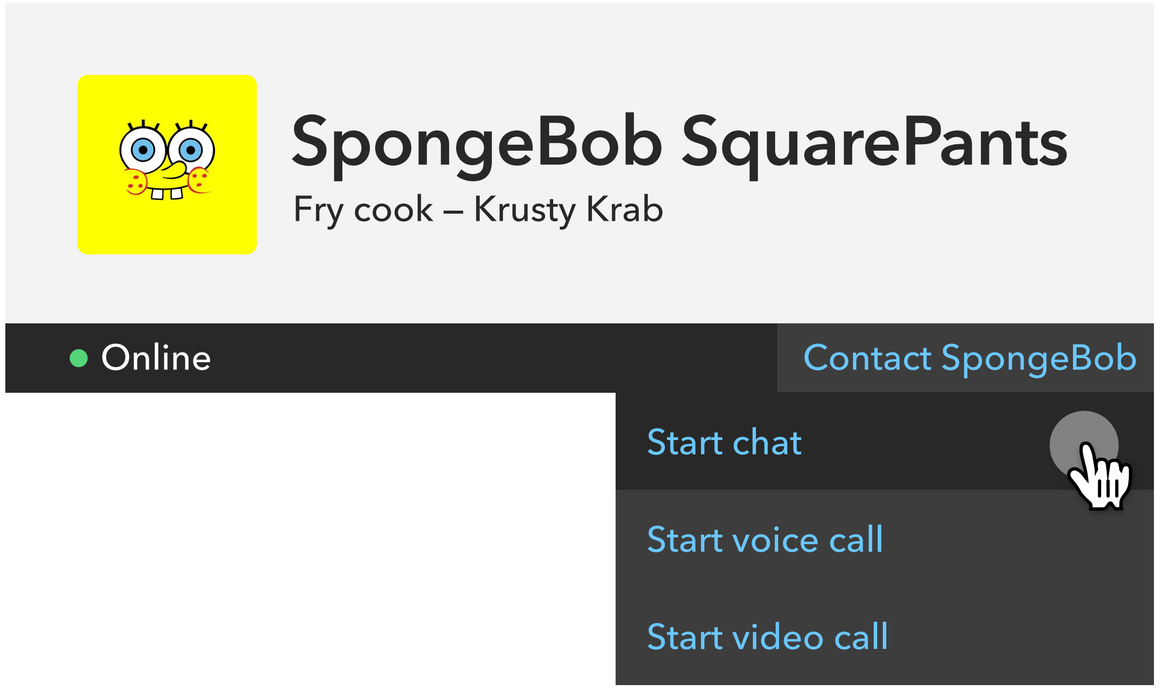
```
import { render, fireEvent } from '@testing-library/react';

test('options are visible when toggle is clicked', () => {
  const { getByText } = render(
    <OverflowMenu toggleText="Open menu">
      <MenuItem>Option 1</MenuItem>
    </OverflowMenu>
  );
  fireEvent.click(getByText('Open menu'));
  expect(getByText('Option 1')).toBeVisible();
});
```

Test component variations

```
const startChat = () => {}

<ContactCard
  {...contactInfo}
  online
  onChat={startChat}
/>
```



```
const startChat = () => {}

<ContactCard
  {...contactInfo}
  online
  onChat={startChat}
/>
```



src/ContactCard/ContactCard.test.js

```
test('initializes chat', () => {
  const onChatMock = jest.fn();
  const { getByText } = render(
     <ContactCard onChat={onChatMock} online {...contact} />
  );

  fireEvent.click(getByText('Contact SpongeBob'));
  fireEvent.click(getByText('Start chat'));

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

src/ContactCard/ContactCard.test.js

```
test('initializes chat', () => {
  const onChatMock = jest.fn();
  const { getByText } = render(
    <ContactCard onChat={onChatMock} online {...contact} />
  );

  fireEvent.click(getByText('Contact SpongeBob'));
  fireEvent.click(getByText('Start chat'));

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

src/ContactCard/ContactCard.test.js

```
test('initializes chat', () => {
  const onChatMock = jest.fn();
  const { getByText } = render(
    <ContactCard onChat={onChatMock} online {...contact} />
  );

  fireEvent.click(getByText('Contact SpongeBob'));
  fireEvent.click(getByText('Start chat'));

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

src/ContactCard/ContactCard.test.js

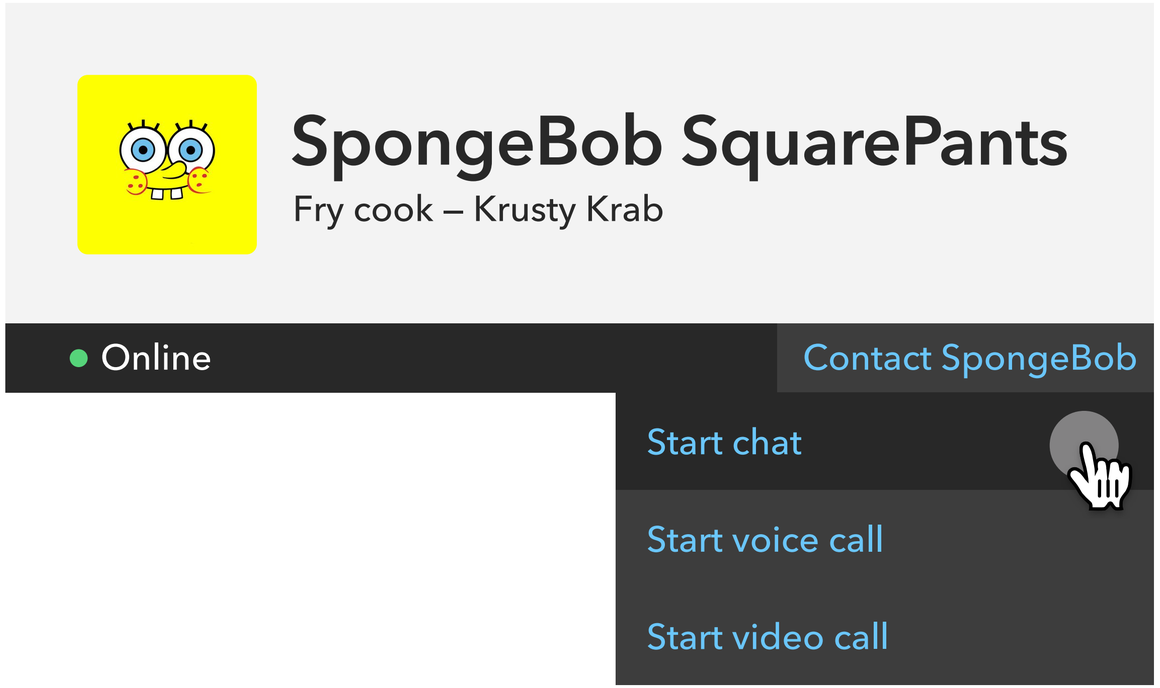
```
test('initializes chat', () => {
  const onChatMock = jest.fn();
  const { getByText } = render(
    <ContactCard onChat={onChatMock} online {...contact} />
  );

  fireEvent.click(getByText('Contact SpongeBob'));
  fireEvent.click(getByText('Start chat'));

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

src/ContactCard/ContactCard.test.js

```
test('initializes chat', () => {
  const onChatMock = jest.fn();
  const { getByText } = render(
     <ContactCard onChat={onChatMock} online {...contact} />
  );

  fireEvent.click(getByText('Contact SpongeBob'));
  fireEvent.click(getByText('Start chat'));

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

The power of integration tests

```
const startChat = () => {}

<ContactCard
  {...contactInfo}
  online
  onChat={startChat}
/>
```



src/ContactCard/ContactCard.test.js

```
test('initializes chat', () => {
  const onChatMock = jest.fn();
  const { getByText } = render(
    <ContactCard onChat={onChatMock} online {...contact} />
  );

  fireEvent.click(getByText('Contact SpongeBob'));
  fireEvent.click(getByText('Start chat'));

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

Why is testing with Jest beneficial?

100s of tests in seconds

100s of tests in seconds

Interact with UI like an end-user

100s of tests in seconds

Interact with UI like an end-user

Test integrations between components

100s of tests in seconds

Interact with UI like an end-user

Test integrations between components

Makes you a valuable developer


Recap

Use Storybook addons

Use UI labels to find DOM elements

Test state and props variations

Make assertions as specific as possible

Write more integration tests

Thank you!