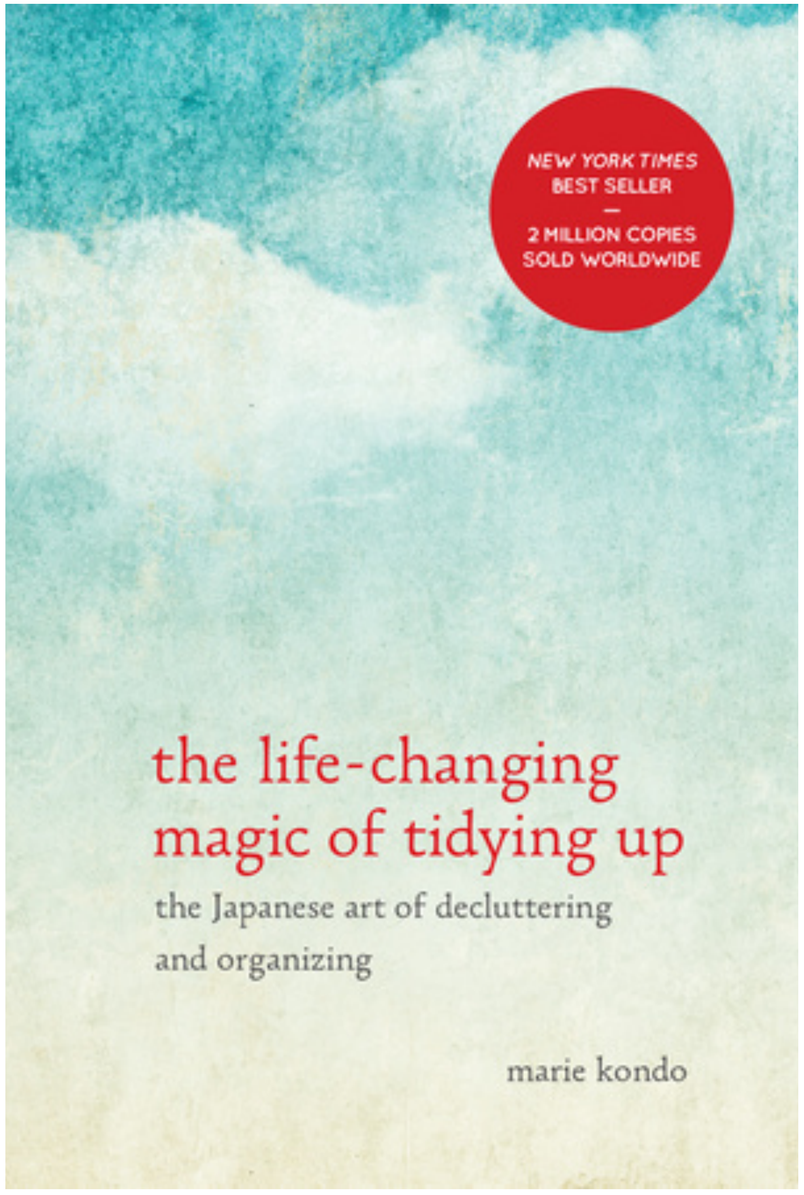
Konmari Your Code

Finding Joy in Refactoring

Reactathon - December 2020

👋 Hi, I'm Becca

Formidable


NEW YORK TIMES
BEST SELLER
—
2 MILLION COPIES
SOLD WORLDWIDE
the life-changing
magic of tidying up
the Japanese art of decluttering
and organizing
marie kondo


STORAGE IS NOT THE ANSWER TO CLUTTER!
YOU MUST BEGIN BY DISCARDING!

What is refactoring?

“Code refactoring is the process of restructuring existing computer code—changing the factoring—without changing its external behavior.”

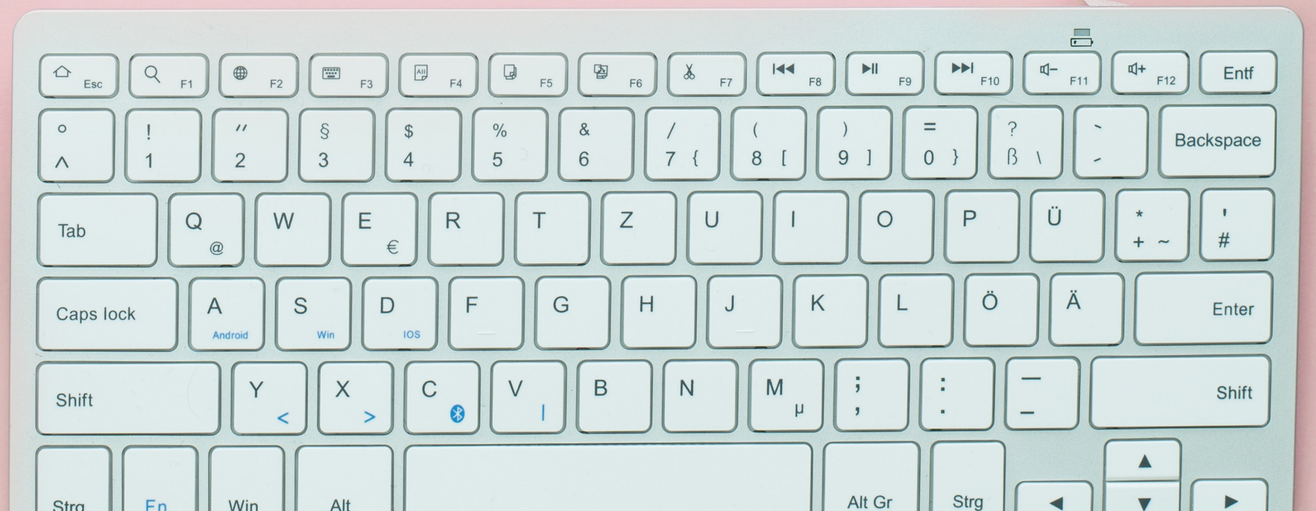
–Martin Fowler

Factoring

Internal parts of a program or implementation details

* Changes to the external-facing API of a library or top-level component is *not* refactoring.

What decisions would I make if I could I make if I were to write this code again?

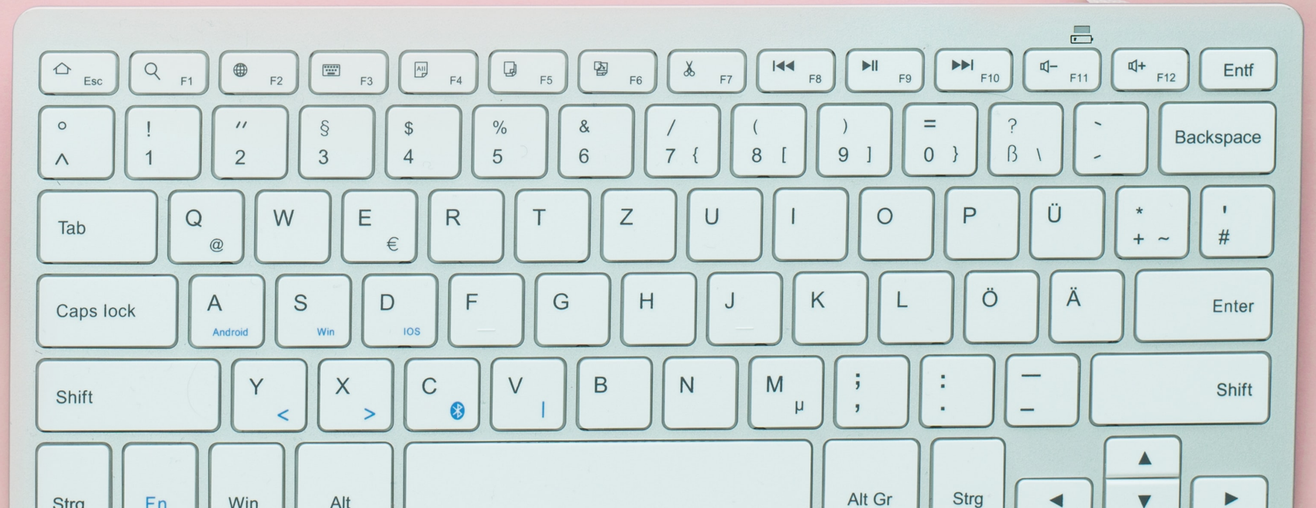


Refactoring to modernize syntax

```tsx
class BaseCard extends React.Component<BaseCardProps, BaseCardState> {
  constructor(props: BaseCardProps) {
    super(props);
    this.state = {
      expanded: props.initialExpanded || false
    };
  }

  handleToggle = () => {
    this.setState(
      ({ expanded }) => ({
        expanded: !expanded
      })
    );
  };

  render() {
    const { title, children } = this.props;
    const { expanded } = this.state;
    return (
      <Card>
        <Card.Header>
          <Card.HeaderContent>{title}</Card.HeaderContent>
          <Card.Toggle expanded={expanded} onToggle={this.handleToggle} />
        </Card.Header>
        <Card.Body visible={expanded}>{children}</Card.Body>
      </Card>
    );
  }
}
```

```tsx
const BaseCard: React.FunctionComponent<BaseCardProps> = ({
  initialExpanded = false,
  title,
  children
}) => {
  const [expanded, setExpanded] = React.useState(initialExpanded);

  function handleToggle() {
    setExpanded(expanded => !expanded);
  }

  return (
    <Card>
      <Card.Header>
        <Card.HeaderContent>{title}</Card.HeaderContent>
        <Card.Toggle expanded={expanded} onToggle={handleToggle} />
      </Card.Header>
      <Card.Body visible={expanded}>{children}</Card.Body>
    </Card>
  );
};
```

When refactoring, **modern syntax is a tool, not an end goal.**

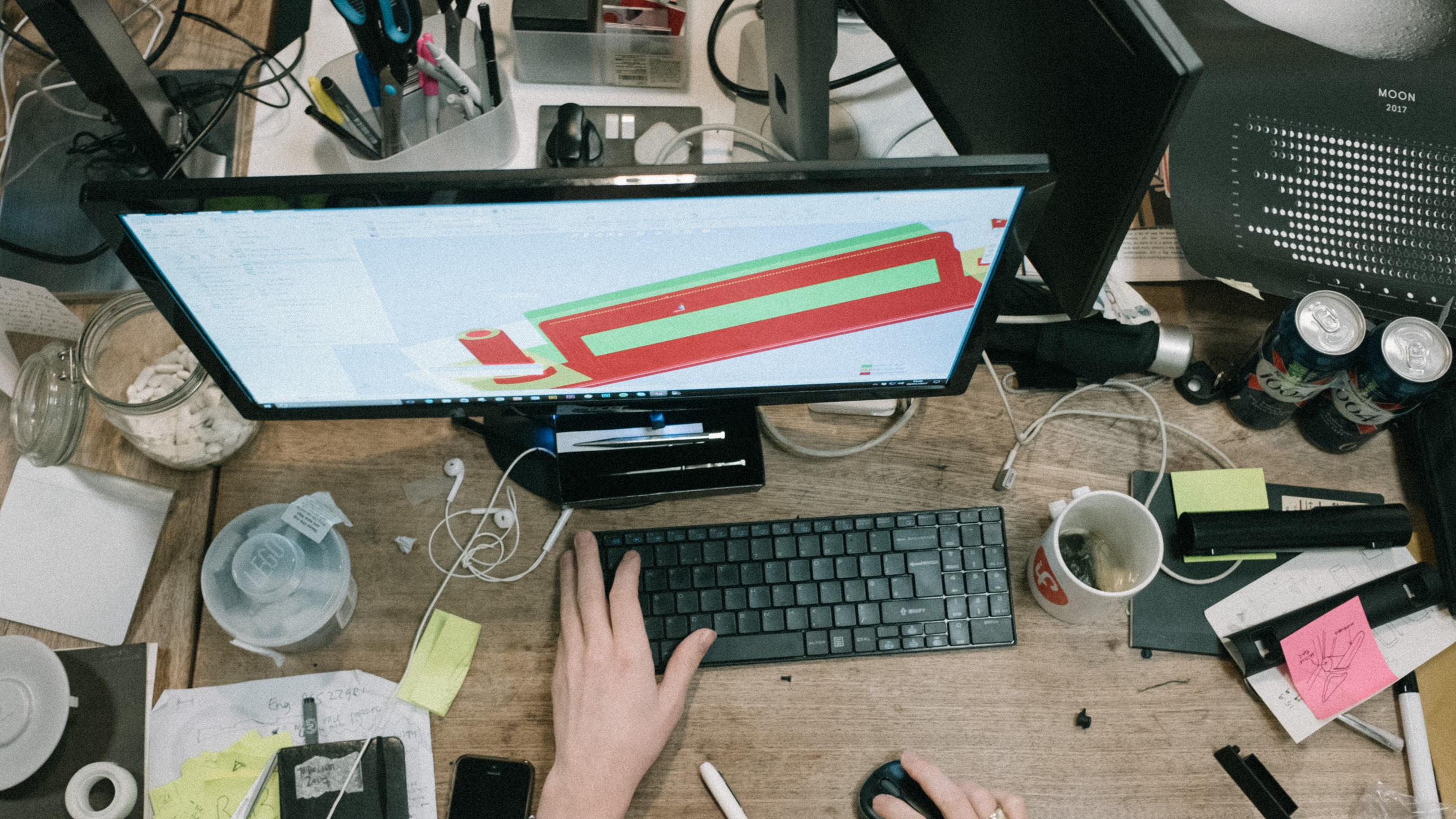


@beccaliz


Refactoring to create
“clean code”
@beccaliz


МИР
LEGO
SPEED
STAR WARS
BRICK HEADZ

When do I refactor?


MOON
2017

Hard to test

Hard to understand

Hard to change

Out-of-control props

Nested conditionals

Confusing variable names

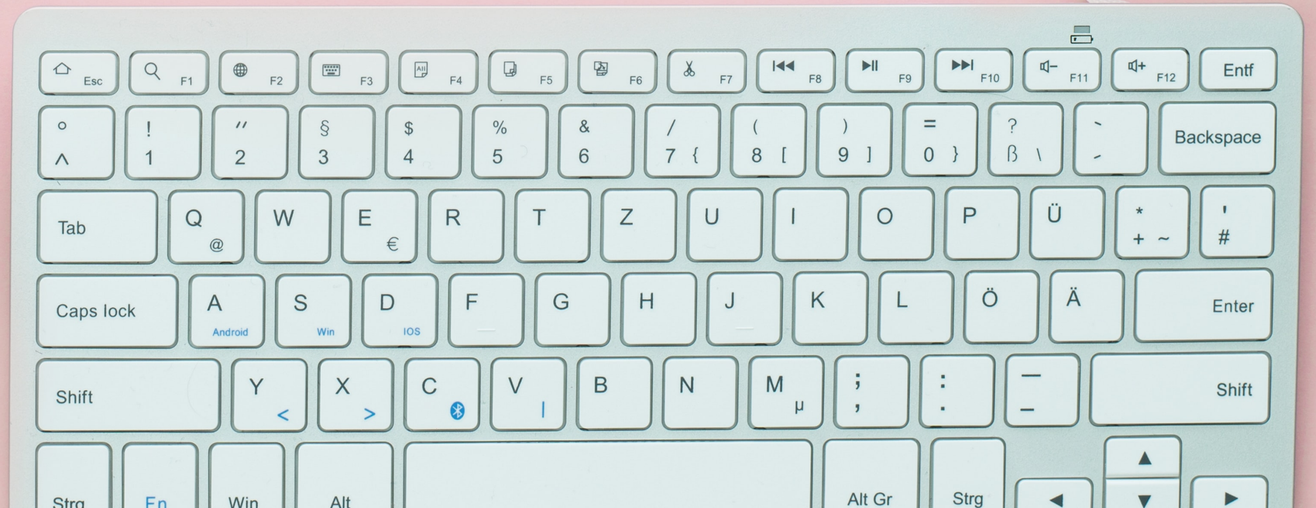
Components doing too many things


Trust yourself
@beccaliz

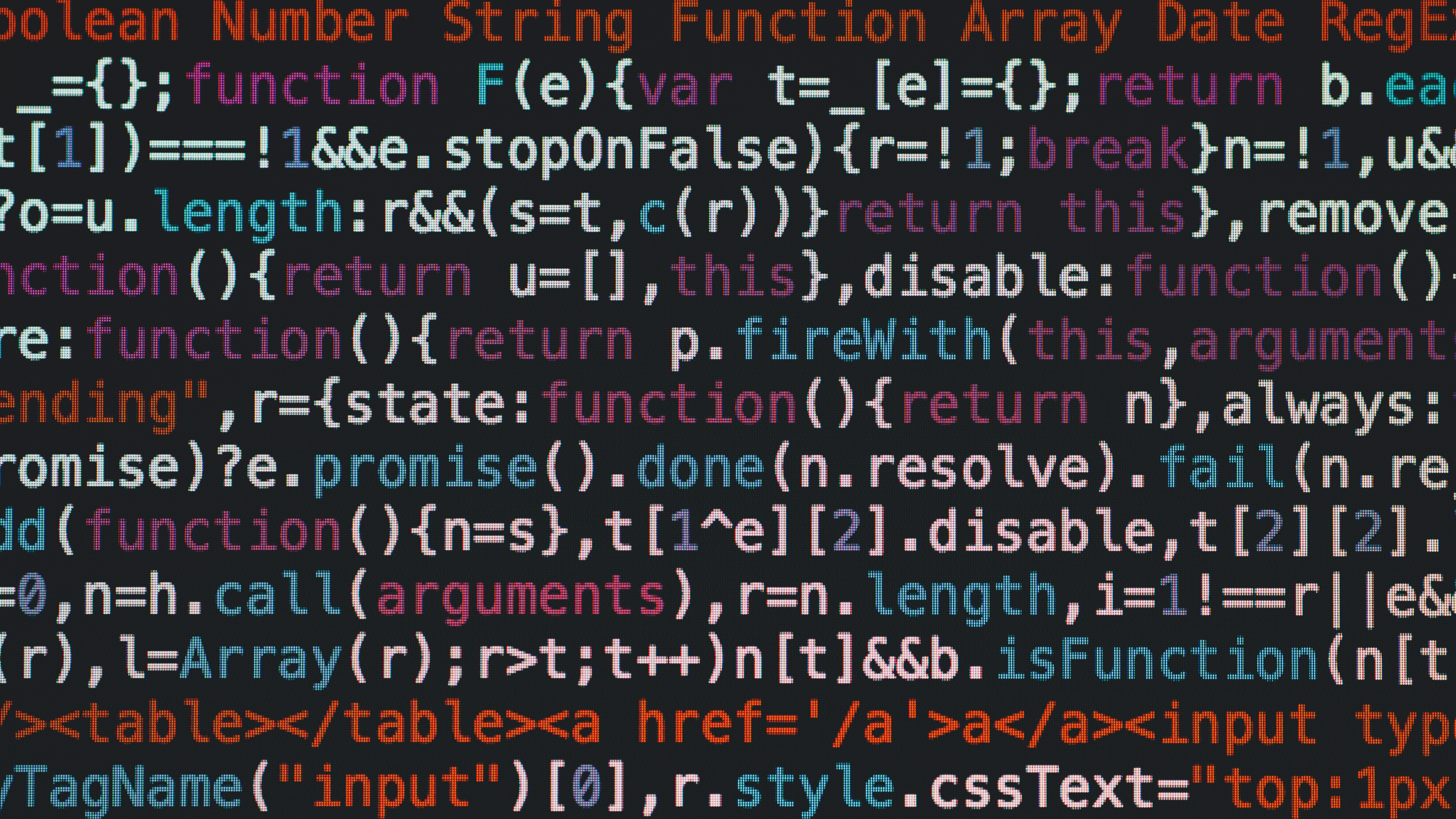

A couple more things
to note…
@beccaliz

`git blame`

Refactoring doesn't mean that you did something wrong.



Discarding

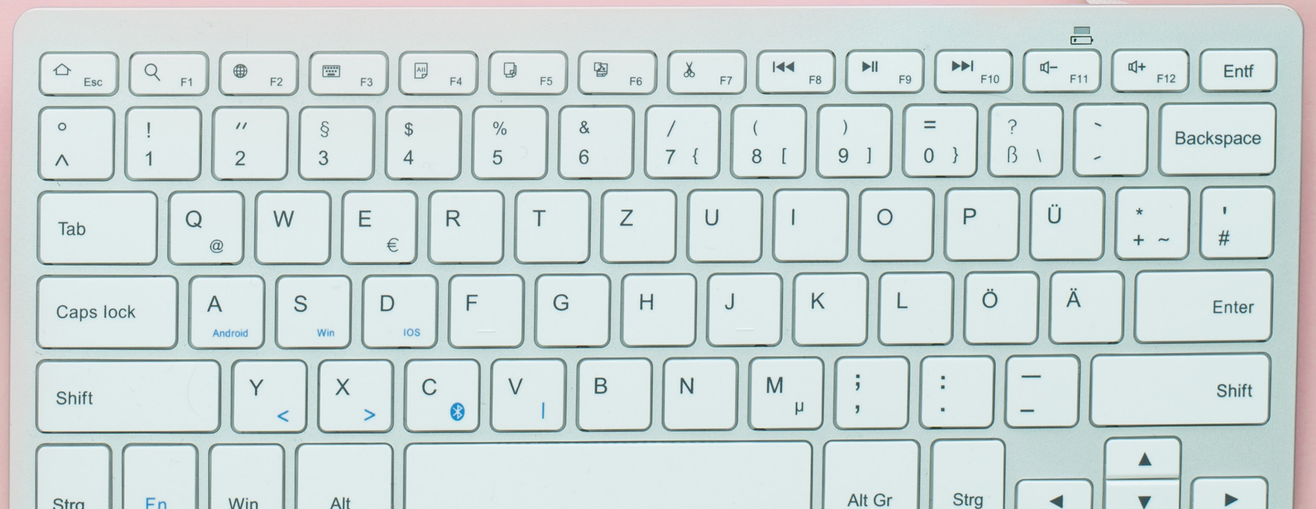

oolean Number String Function Array Date RegE
={};function F(e){var t=[e]={};return b.ea
t[1])===!1&&e.stopOnFalse){r=!1;break}n=!1,u&
?o=u.length:r&&(s=t,c(r))}return this},remove
nction(){return u=[],this},disable:function()
re:function(){return p.fireWith(this,argument
ending",r={state:function(){return n},always:
romise)?e.promise().done(n.resolve).fail(n.re
dd(function(){n=s},t[1^e][2].disable,t[2][2].
=0,n=h.call(arguments),r=n.length,i=1!==r||e&
(r),l=Array(r);r>t;t++)n[t]&&b.isFunction(n[t
/><table></table>a<input typ
yTagName("input")[0],r.style.cssText="top:1px

```
git checkout .
```

Pain Points

Testing

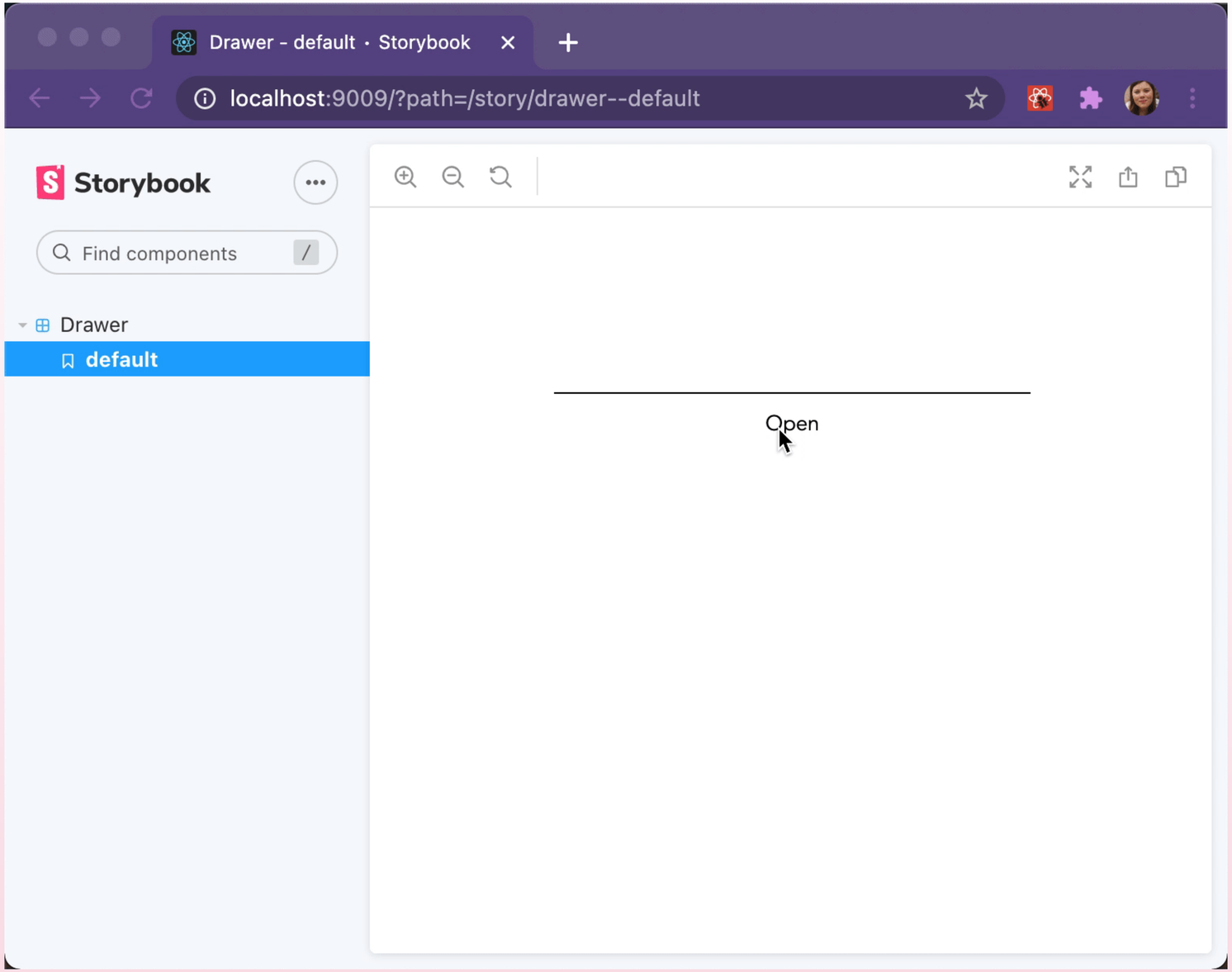
Not all tests are made alike



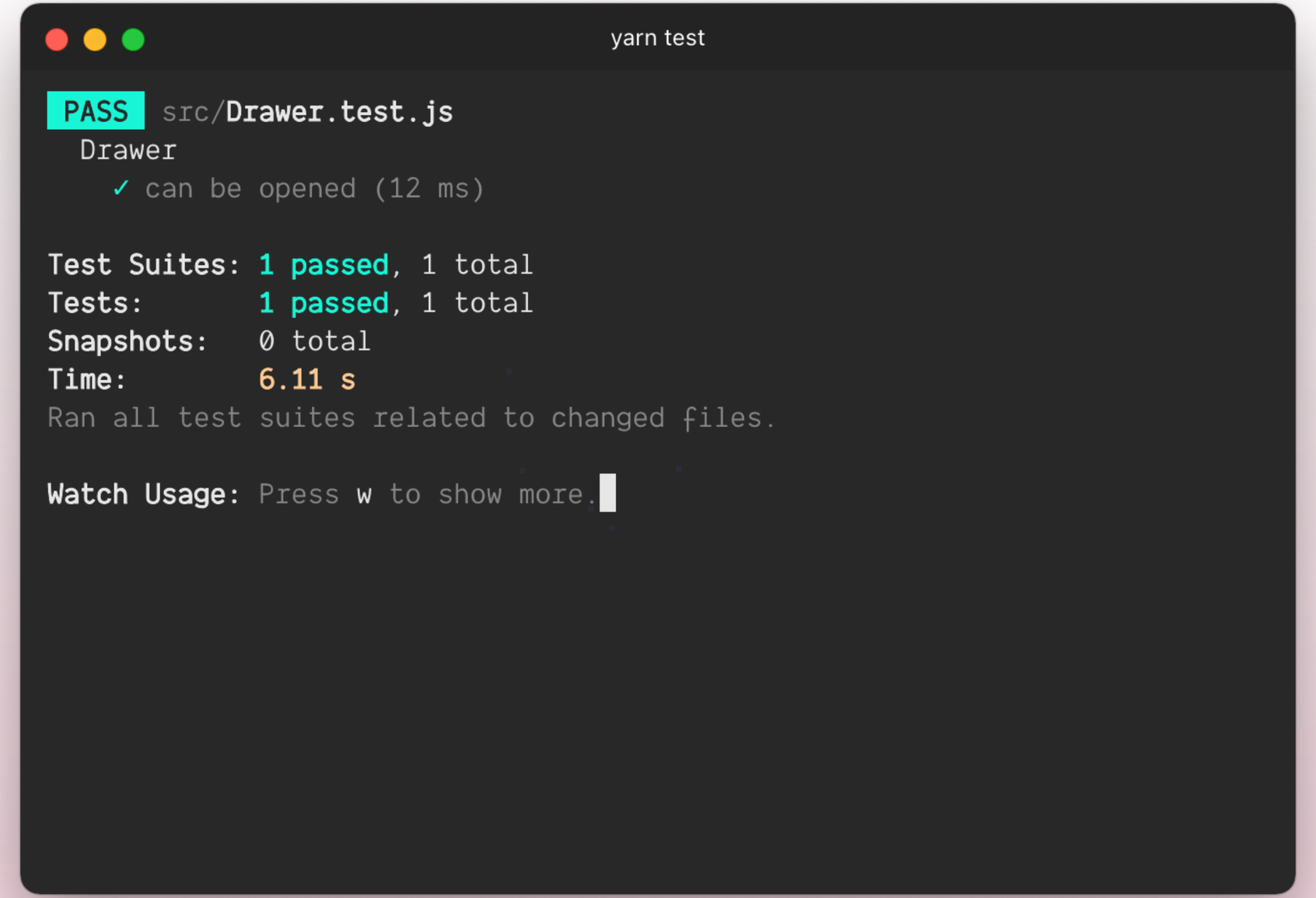
```
export class Drawer extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      open: props.initialOpen || false,
    };
  }

  render() {
    return (
      <div
        className={`drawer ${
          this.state.open ? "drawer--open" : "drawer--closed"
        }`}
      >
        {this.state.open && (
          <div className="drawer__contents">{this.props.children}</div>
        )}
        <button
          className="drawer__toggle"
          onClick={() => this.setState((state) => ({ open: !state.open }))}
        >
          {this.state.open ? "Close" : "Open"}
        </button>
      </div>
```

```
export class Drawer extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      open: props.initialOpen || false,
    };
  }

  render() {
    return (
      <div
        className={`drawer ${
          this.state.open ? "drawer--open" : "drawer--closed"
        }`}
      >
        {this.state.open && (
          <div className="drawer__contents">{this.props.children}</div>
        )}
        <button
          className="drawer__toggle"
          onClick={() => this.setState((state) => ({ open: !state.open }))}
        >
          {this.state.open ? "Close" : "Open"}
        </button>
      </div>
```

```
    super(props);
    this.state = {
      open: props.initialOpen || false,
    };
  }

  render() {
    return (
      <div
        className={`drawer ${
          this.state.open ? "drawer--open" : "drawer--closed"
        }`}
      >
        {this.state.open && (
          <div className="drawer__contents">{this.props.children}</div>
        )}
        <button
          className="drawer__toggle"
          onClick={() => this.setState((state) => ({ open: !state.open }))}
        >
          {this.state.open ? "Close" : "Open"}
        </button>
      </div>
    );
  }
}
```


Drawer - default · Storybook
localhost:9009/?path=/story/drawer--default
Storybook
Find components
/
Drawer
default
Open

```
it("can be opened", () => {
  const wrapper = shallow(<Drawer>Some content</Drawer>);
  const toggle = wrapper.find("button");

  toggle.props().onClick();

  expect(wrapper.state("open")).toBe(true);
});
```

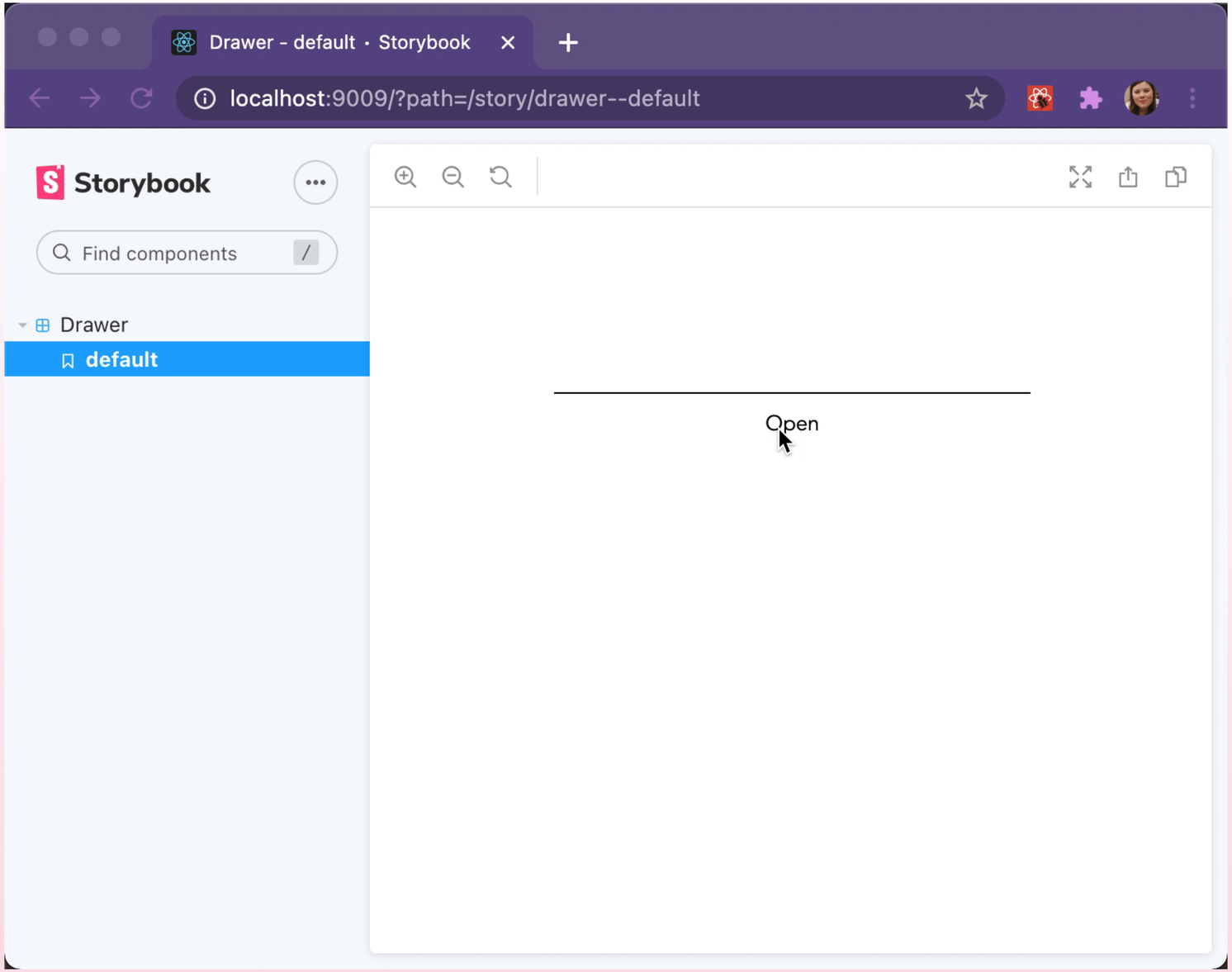

yarn test
PASS src/Drawer.test.js
Drawer
✓ can be opened (12 ms)
Test Suites: 1 passed, 1 total
Tests: 1 passed, 1 total
Snapshots: 0 total
Time: 6.11 s
Ran all test suites related to changed files.
Watch Usage: Press w to show more.

```
export class Drawer extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      open: props.initialOpen || false,
    };
  }

  render() {
    return (
      <div
        className={`drawer ${
          this.state.open ? "drawer--open" : "drawer--closed"
        }`}
      >
        {this.state.open && (
          <div className="drawer__contents">{this.props.children}</div>
        )}
        <button
          className="drawer__toggle"
          onClick={() => this.setState((state) => ({ open: !state.open }))}
        >
          {this.state.open ? "Close" : "Open"}
        </button>
      </div>
```

```
export const Drawer = ({ children, initialOpen = false }) => {
  const [open, setOpen] = React.useState(initialOpen);

  return (
    <div className={`drawer ${open ? "drawer--open" : "drawer--closed"}`}>
      {open && <div className="drawer__contents">{children}</div>}
      <button className="drawer__toggle" onClick={() => setOpen(!open)}>
        {open ? "Close" : "Open"}
      </button>
    </div>
  );
};
```


yarn test
FAIL src/Drawer.test.js
Drawer
× can be opened (9 ms)
● Drawer › can be opened
ShallowWrapper::state() can only be called on class components
9 | toggle.props().onClick();
10 |
> 11 | expect(wrapper.state("open")).toBe(true);
^
12 | });
13 | });
14 |
at ShallowWrapper.state (node_modules/enzyme/src/ShallowWrapper.js:1192:13)
at Object.<anonymous> (src/Drawer.test.js:11:20)

Behavior vs. implementation details

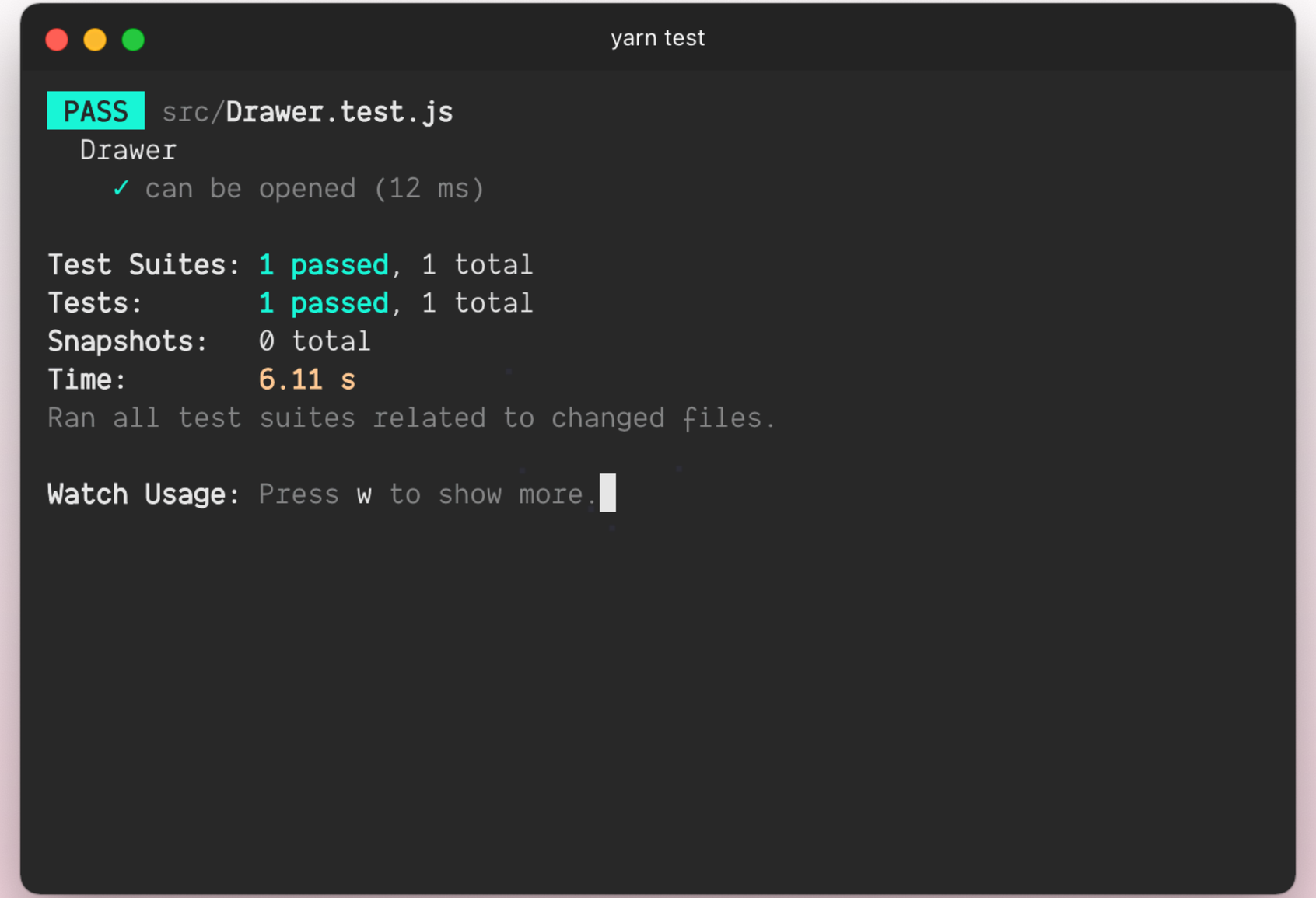

Drawer - default · Storybook
localhost:9009/?path=/story/drawer--default
Storybook
Find components
/
Drawer
default
Open

```tsx
it("can be opened", () => {
  const { getByText, container } = render(<Drawer>Some content</Drawer>);

  fireEvent.click(getByText("Open"));

  expect(container.innerHTML).toContain("Some content");
});
```

```tsx
it("can be opened", () => {
  const { getByText, container } = render(<Drawer>Some content</Drawer>);

  fireEvent.click(getByText("Open"));

  expect(container.innerHTML).toContain("Some content");
});
```

```tsx
it("can be opened", () => {
  const { getByText, container } = render(<Drawer>Some content</Drawer>);

  fireEvent.click(getByText("Open"));

  expect(container.innerHTML).toContain("Some content");
});
```

```
it("can be opened", () => {
  const { getByText, container } = render(<Drawer>Some content</Drawer>);

  fireEvent.click(getByText("Open"));

  expect(container.innerHTML).toContain("Some content");
});
```


yarn test
PASS src/Drawer.test.js
Drawer
✓ can be opened (12 ms)
Test Suites: 1 passed, 1 total
Tests: 1 passed, 1 total
Snapshots: 0 total
Time: 6.11 s
Ran all test suites related to changed files.
Watch Usage: Press w to show more.

Incremental Changes

```
export const Drawer = ({ children, initialOpen = false }) => {
  const [open, setOpen] = React.useState(initialOpen);

  return (
    <div className={`drawer ${open ? "drawer--open" : "drawer--closed"}`}>
      {open && <div className="drawer__contents">{children}</div>}
      <button className="drawer__toggle" onClick={() => setOpen(!open)}>
        {open ? "Close" : "Open"}
      </button>
    </div>
  );
};
```

```jsx
export const Drawer = ({ children, initialOpen = false }) => {
  const [open, setOpen] = React.useState(initialOpen);

  if (open) {
    return (
      <div className={`drawer ${open ? "drawer--open" : "drawer--closed"}`}>
        {open && <div className="drawer__contents">{children}</div>}
        <button className="drawer__toggle" onClick={() => setOpen(!open)}>
          {open ? "Close" : "Open"}
        </button>
      </div>
    );
  }
  return (
    <div className={`drawer ${open ? "drawer--open" : "drawer--closed"}`}>
      {open && <div className="drawer__contents">{children}</div>}
      <button className="drawer__toggle" onClick={() => setOpen(!open)}>
        {open ? "Close" : "Open"}
      </button>
    </div>
  );
};
```

```jsx
export const Drawer = ({ children, initialOpen = false }) => {
  const [open, setOpen] = React.useState(initialOpen);

  if (open) {
    return (
      <div className="drawer--open">
        {open && <div className="drawer__contents">{children}</div>}
        <button className="drawer__toggle" onClick={() => setOpen(!open)}>
          {open ? "Close" : "Open"}
        </button>
      </div>
    );
  }
  return (
    <div className="drawer--closed">
      {open && <div className="drawer__contents">{children}</div>}
      <button className="drawer__toggle" onClick={() => setOpen(!open)}>
        {open ? "Close" : "Open"}
      </button>
    </div>
  );
};
```

```jsx
export const Drawer = ({ children, initialOpen = false }) => {
  const [open, setOpen] = React.useState(initialOpen);

  if (open) {
    return (
      <div className="drawer--open">
        <div className="drawer__contents">{children}</div>
        <button className="drawer__toggle" onClick={() => setOpen(!open)}>
          {open ? "Close" : "Open"}
        </button>
      </div>
    );
  }
  return (
    <div className="drawer--closed">
     <button className="drawer__toggle" onClick={() => setOpen(!open)}>
        {open ? "Close" : "Open"}
      </button>
    </div>
  );
};
```

```jsx
export const Drawer = ({ children, initialOpen = false }) => {
  const [open, setOpen] = React.useState(initialOpen);

  if (open) {
    return (
      <div className="drawer--open">
        <div className="drawer__contents">{children}</div>
        <button className="drawer__toggle" onClick={() => setOpen(false)}>
          Close
        </button>
      </div>
    );
  }
  return (
    <div className="drawer--closed">
     <button className="drawer__toggle" onClick={() => setOpen(true)}>
        Open
      </button>
    </div>
  );
};
```

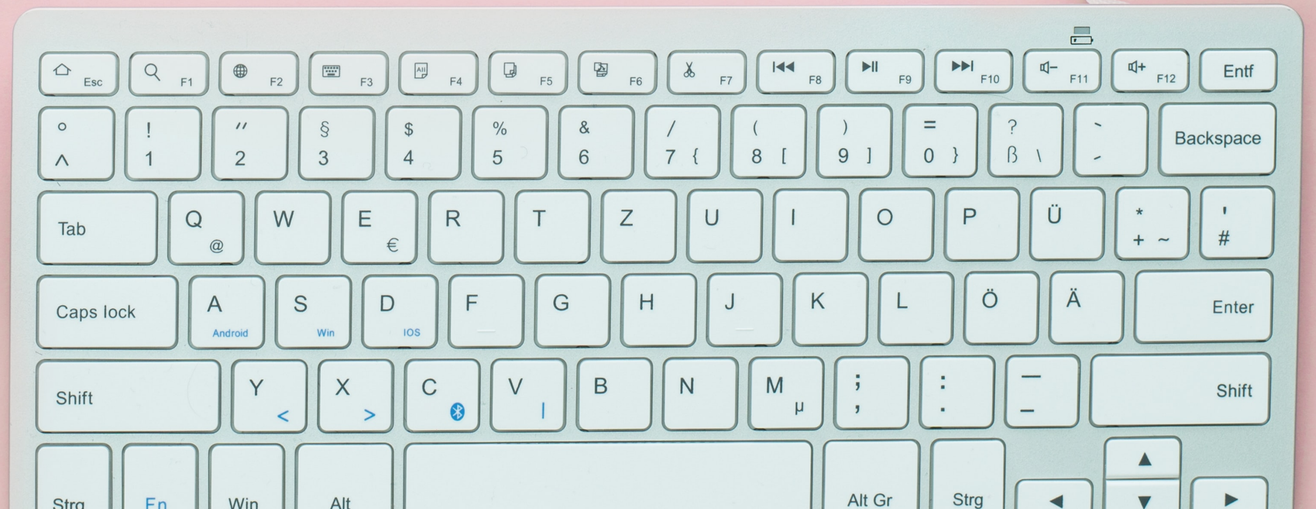

Putting things away
@beccaliz

```
/components
  /dresser
    /dresser.js
    /dresser.test.js
    /drawer.js
  /cabinet
    /cabinet.js
    /cabinet.test.js
    /door.js
  /shared
    /handle.js
```

Refactoring as a daily practice

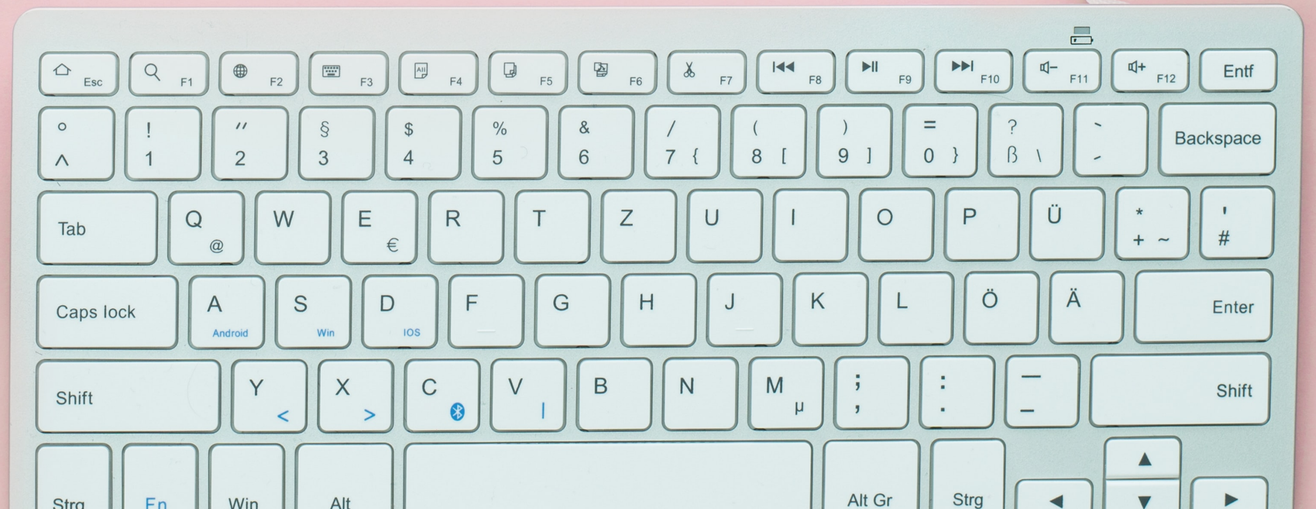
Working with time constraints

Making refactoring a part of our daily routine



How do I get **other people** on board?

Consistency is key



@beccaliz

JS Style Guide

Testing

Testing is a really important part of component design. Not only can it help catch bugs, but it can help document our code and inform our decisions about the component or function's API. If a component or function is hard to test, then it might be time to re-think the way it is designed.

Libraries

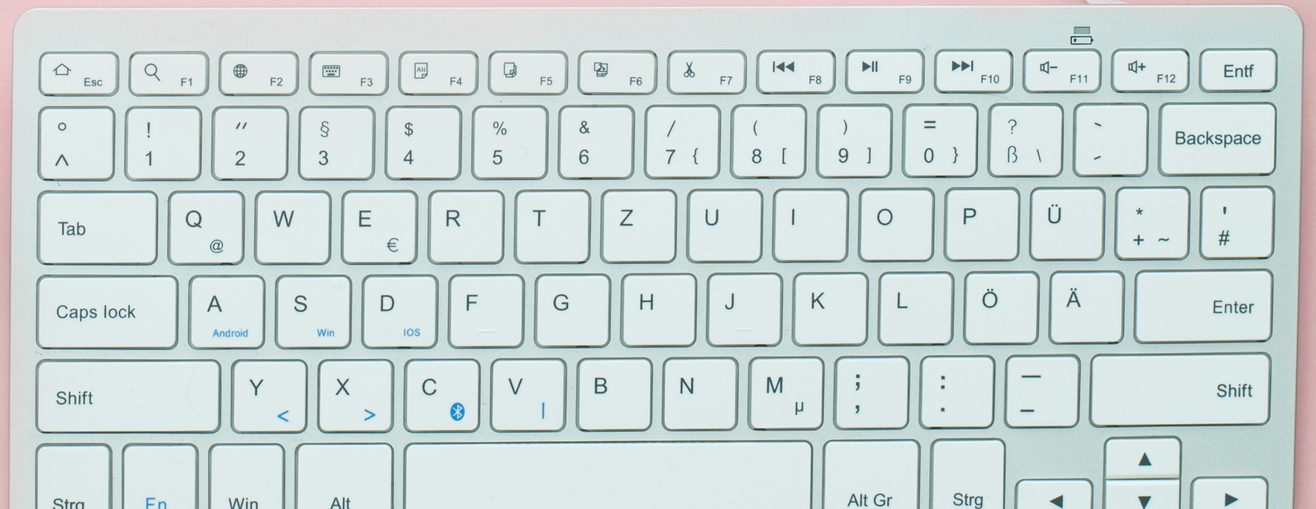
We are using Jest and React Testing Library. We also have legacy tests that use Enzyme and React Test Renderer. If you are updating a component that uses one of these test renderers, please update the tests to use React Testing Library.

Test behavior, not implementation details

React Testing Library is designed to test behavior over implementation. This means that we are testing the component in the way that a user would interact with it.

Tests that depend on implementation details such as the component's internal state or helper functions are brittle, and don't give us the assurance we need that the component actually behaves as expected for a user. Ideally, we should be able to refactor our code without changing our tests.

The end goal of refactoring is to better understand your code.



@beccaliz


Gratitude for the work
that has already been
done
@beccaliz

“You’ll be surprised at how many of the things you possess have already fulfilled their role. By acknowledging their contribution and letting them go with gratitude, you will be able to truly put the things you own, and your life, in order.”

–Marie Kondo

Thank you!

@beccaliz

becca.is