The Wrapper Object

In this lesson, we will learn about the main object of `vue-test-utils: the Wrapper Object.

WE'LL COVER THE FOLLOWING

So far, we've made tests using Jest Snapshots. In most cases, that's what we'll use, but sometimes we want to assert something more specific.

Although you can access the Vue instance via `cmp.vm`, you have a set of utilities at your disposal to make it easier. Let's see what we can do.

The Wrapper Object

The `Wrapper` is the main object of `vue-test-utils`. It is the type returned by mount, shallowMount, find and findAll functions.

`find` and `findAll`

They accept a `selector` as an argument, which can be both a CSS selector or a Vue Component.

So we can do things like:

```
let cmp = mount(MessageList);
expect(cmp.find(".message").element).toBeInstanceOf(HTMLElement);

// Or even call it multiple times
let el = cmp.find(".message").find("span").element;

// Although, the previous line could also be written in one call
let el = cmp.find(".message span").element;
```

Asserting Structure

Let's add more tests to `MessageList.test.js`:

MessageList.test.js

```
import { mount } from '@vue/test-utils'
import MessageList from '../src/components/MessageList'
import Message from '../src/components/Message'

describe('MessageList.test.js', () => {
    let cmp

    beforeEach(() => {
      cmp = mount(MessageList, {
        // Be aware that props is overridden using `propsData`
        propsData: {
          messages: ['Cat']
        }
      })
    })

    it('has received ["Cat"] as the message property', () => {
      expect(cmp.props().messages).toEqual(['Cat'])
    })

    it('has the expected html structure', () => {
      expect(cmp.element).toMatchSnapshot()
    })

    it("is a MessageList component", () => {
      expect(cmp.is(MessageList)).toBe(true);

      // Or with CSS selector
      expect(cmp.is("ul")).toBe(true);
    });

    it("contains a Message component", () => {
      expect(cmp.contains(Message)).toBe(true);

      // Or with CSS selector
      expect(cmp.contains(".message")).toBe(true);
    });
```

Here, we're using `is` to assert the root component type, and `contains` to check for the existence of sub-components. Just like `find` they receive a Selector, which can be a CSS Selector or a Component.

We have some utils to assert the **Vue instance**:

MessageList.test.js

```
import { mount } from '@vue/test-utils'
```

```
import MessageList from '../src/components/MessageList'
import Message from '../src/components/Message'

describe('MessageList.test.js', () => {
    let cmp

    beforeEach(() => {
      cmp = mount(MessageList, {
        // Be aware that props is overridden using `propsData`
        propsData: {
          messages: ['Cat']
        }
      })
    })

    it('has received ["Cat"] as the message property', () => {
      expect(cmp.props().messages).toEqual(['Cat'])
    })

    it('has the expected html structure', () => {
      expect(cmp.element).toMatchSnapshot()
    })

    it("is a MessageList component", () => {
      expect(cmp.is(MessageList)).toBe(true);

      // Or with CSS selector
      expect(cmp.is("ul")).toBe(true);
    });

    it("contains a Message component", () => {
      expect(cmp.contains(Message)).toBe(true);

      // Or with CSS selector
      expect(cmp.contains(".message")).toBe(true);
    });

    it("Both MessageList and Message are vue instances", () => {
    expect(cmp.isVueInstance()).toBe(true);
    expect(cmp.find(Message).isVueInstance()).toBe(true);
  });
```

Now we're going to assert the **Structure** in more detail:

MessageList.test.js

```
import { mount } from '@vue/test-utils'
import MessageList from '../src/components/MessageList'
import Message from '../src/components/Message'

describe('MessageList.test.js', () => {
    let cmp

    beforeEach(() => {
      cmp = mount(MessageList, {
        // Be aware that props is overridden using `propsData`
        propsData: {
          messages: ['Cat']
```

```
    }
  })
})


it('has received ["Cat"] as the message property', () => {
  expect(cmp.props().messages).toEqual(['Cat'])
})

it('has the expected html structure', () => {
  expect(cmp.element).toMatchSnapshot()
})

it("is a MessageList component", () => {
  expect(cmp.is(MessageList)).toBe(true);

  // Or with CSS selector
  expect(cmp.is("ul")).toBe(true);
});

it("contains a Message component", () => {
  expect(cmp.contains(Message)).toBe(true);

  // Or with CSS selector
  expect(cmp.contains(".message")).toBe(true);
});

it("Both MessageList and Message are vue instances", () => {
  expect(cmp.isVueInstance()).toBe(true);
  expect(cmp.find(Message).isVueInstance()).toBe(true);
});

it("Message element exists", () => {
expect(cmp.find(".message").exists()).toBe(true);
  });

it("Message is not empty", () => {
  expect(cmp.find(Message).isEmpty()).toBe(false);
});

it('Message has a class attribute set to "message"', () => {
  expect(cmp.find(Message).attributes().class).toBe("message");
});
```

The exists, isEmpty and attributes methods come in handy for this.

Asserting Style

Then, we have `classes` and `attributes().style` to assert **styling**. Let's update the `Message.vue` component with a style, since `attributes().style` asserts only inline styles:

Message.vue

```
<li style="margin-top: 10px" class="message">{{message}}</li>
```

Now, let's add the tests:

MessageList.test.js

```
import { mount } from '@vue/test-utils'
import MessageList from '../src/components/MessageList'
import Message from '../src/components/Message'

describe('MessageList.test.js', () => {
    let cmp

    beforeEach(() => {
      cmp = mount(MessageList, {
        // Be aware that props is overridden using `propsData`
        propsData: {
          messages: ['Cat']
        }
      })
    })

    it('has received ["Cat"] as the message property', () => {
      expect(cmp.props().messages).toEqual(['Cat'])
    })

    it('has the expected html structure', () => {
      expect(cmp.element).toMatchSnapshot()
    })

    it("is a MessageList component", () => {
      expect(cmp.is(MessageList)).toBe(true);

      // Or with CSS selector
      expect(cmp.is("ul")).toBe(true);
    });

    it("contains a Message component", () => {
      expect(cmp.contains(Message)).toBe(true);

      // Or with CSS selector
      expect(cmp.contains(".message")).toBe(true);
    });

    it("Both MessageList and Message are vue instances", () => {
      expect(cmp.isVueInstance()).toBe(true);
      expect(cmp.find(Message).isVueInstance()).toBe(true);
    });

    it("Message element exists", () => {
    expect(cmp.find(".message").exists()).toBe(true);
      });

    it("Message is not empty", () => {
      expect(cmp.find(Message).isEmpty()).toBe(false);
    });

    it('Message has a class attribute set to "message"', () => {
      expect(cmp.find(Message).attributes().class).toBe("message");
    });
    // Style
    it("Message component has the .message class", () => {
```

```
  it("Message component has the .message class", () => {
    expect(cmp.find(Message).classes()).toContain("message");
  });

  it("Message component has style padding-top: 10", () => {
    expect(cmp.find(Message).attributes().style).toBe("padding-top: 10px;");
  });
```

Let’s test a running project of what we have done so far in the next lesson.