Lightning Talks #1

Lightning Talks ⚡

Vue 3 state management

- Why are we recommending a different state management package?

```ts
// * Defining a store JS
export const useTodoJSStore = defineStore("todo", {
    state: () => ({
        todos: new Array(),
    }),
});

// * Defining a store TS
interface State {
    todos: Todo[];
}

class Todo {
    id: number;
    title: string;
    content: string;
    isComplete: boolean;
    constructor(title: string, content: string) {
        this.id = Math.floor(Math.random() * 1337);
        this.title = title;
        this.content = content;
        this.isComplete = false;
    }
}

export const useTodoTSStore = defineStore("todo", {
    state: (): State => ({
        todos: [],
    }),
});
```

1

```ts
export const useTodoTSStore = defineStore("todo", {
    state: (): State => ({
        todos: [],
    }),
    getters: {
        allTodosDone: (state: State): boolean => {
            return state.todos.every(todo => todo.isComplete);
        }
    }
});
```

2

```ts
export const useTodoTSStore = defineStore("todo", {
    state: (): State => ({
        todos: [],
    }),
    getters: {
        allTodosDone: (state: State): boolean => {
            return state.todos.every(todo => todo.isComplete);
        }
    },
    actions: {
        setTodoAsComplete(todoId: number) {
            const todoIndex = this.$state.todos.findIndex(todo => todo.id === todoId);
            if (todoIndex >= 0) this.$state.todos[todoIndex].isComplete = true;
        }
    }
});
```

3

```vue
<script>
    import { mapState } from "pinia";
    import { pinia } from "@/store/store";
    import { useTodoTSStore } from "@/store/modules/todo";

    export default {
        computed: {
            ...mapState(useTodoTSStore, ["todos"]),
        },
        mounted() {
            const todoTSStore = useTodoTSStore(pinia);
            console.log(todoTSStore.$state.todos);
            todoTSStore.$patch((state) => {
                state.todos = new Array();
            });
        }
    }
</script>
```

1

```vue
<script>
    import { mapState, mapActions } from "pinia";
    import { useTodoTSStore } from "@/store/modules/todo";

    export default {
        methods: {
            ...mapActions(useTodoTSStore, ["setTodoAsComplete"])
        },
        computed: {
            ...mapState(useTodoTSStore, ["todos"]),
        },
    }
</script>
```

2

Why you should keep a developer diary

Rickard

The testing trophy

Testing JavaScript according to Kent C. Dodds

Simon

THE FOUR TYPES OF TESTS

End to End

A helper robot that behaves like a user to click around the app and verify that it functions correctly.

Sometimes called “functional testing” or e2e.

Integration

Verify that several units work together in harmony.

Unit

Verify that individual, isolated parts work as expected.

Static

Catch typos and type errors as you write the code.

Let's talk trade-offs

Big problems

Cheap

THE FOUR TYPES OF TESTING

End to End

A helper robot that behaves like a user to click around the app and verify that it functions correctly.

Sometimes called "functional testing" or e2e.

Integration

Verify that several units work together in harmony.

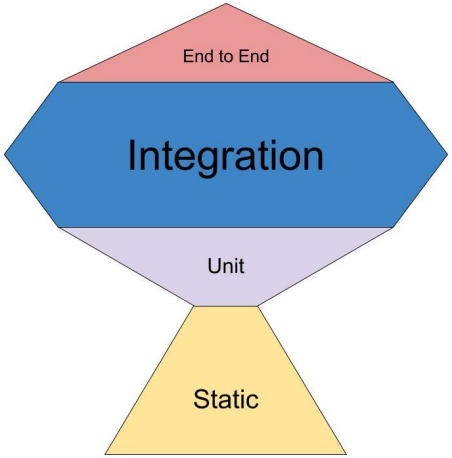
Unit

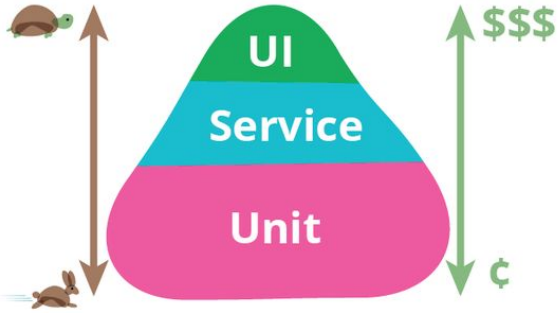
Verify that individual, isolated parts work as expected.

Static

Catch typos and type errors as you write the code.

Simple problems 👌





Write tests. Not too many. Mostly integration.

Getting up to speed with a new project

William

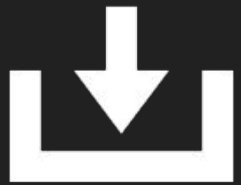
If you know beforehand you'll join the project

- If you have the time and need, try to briefly read up
- Do you know what the project is about and the company behind it?
- What tech is used? Anything you haven't touched before?

Get access to the repos

- Don't let access hinder you later on, ask right away
- Make sure you can checkout the code

But don't get ahead of yourself...



Who's who, What’s what?

- Who owns the project, and why?
- What value does the system aim to create?
- Who is the intended end-user - if any?
- Etc

Get to know the system!

- Get a login for the system if needed
- Ask someone to show you how it works
- Preferably an actual user

If you don't know how the system is intended to work it'll be hard to contribute

Get into the repos

- Look for readme files and/or wikis
- These should help you set up your local environment
- If there is no how-to, create one as you go along
- Ask when stuck!

Get your first ticket

- Preferably small ticket or contained in a specific part of the system
- If you feel unsure, get someone to pair program with you

Contain your excitement

- Learn the part you need to touch for the moment
- Expand your knowledge later on

Code style

- Forget best practice for a moment, and look at how stuff's been done before
- Try to replicate flows and patterns

It is easier to maintain a consistent code base

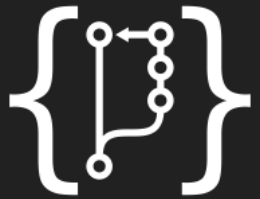
Take notes of any general improvements for later on, when you're more accustomed with the project

PR

- Look up how the PR-strategy for your team works

- Get feedback from teammates and help you improve your understanding of the code

You have now made your first contribution!



In short - thanks for listening!

Learn the value before the code

Get into the code step by step

Look up patterns and flows, mimic

If you're stuck, ask! Use your teammates