CONCISO.

Automatisierte Vue 3 Migration

Dortmund, 28.02.2024

Agenda

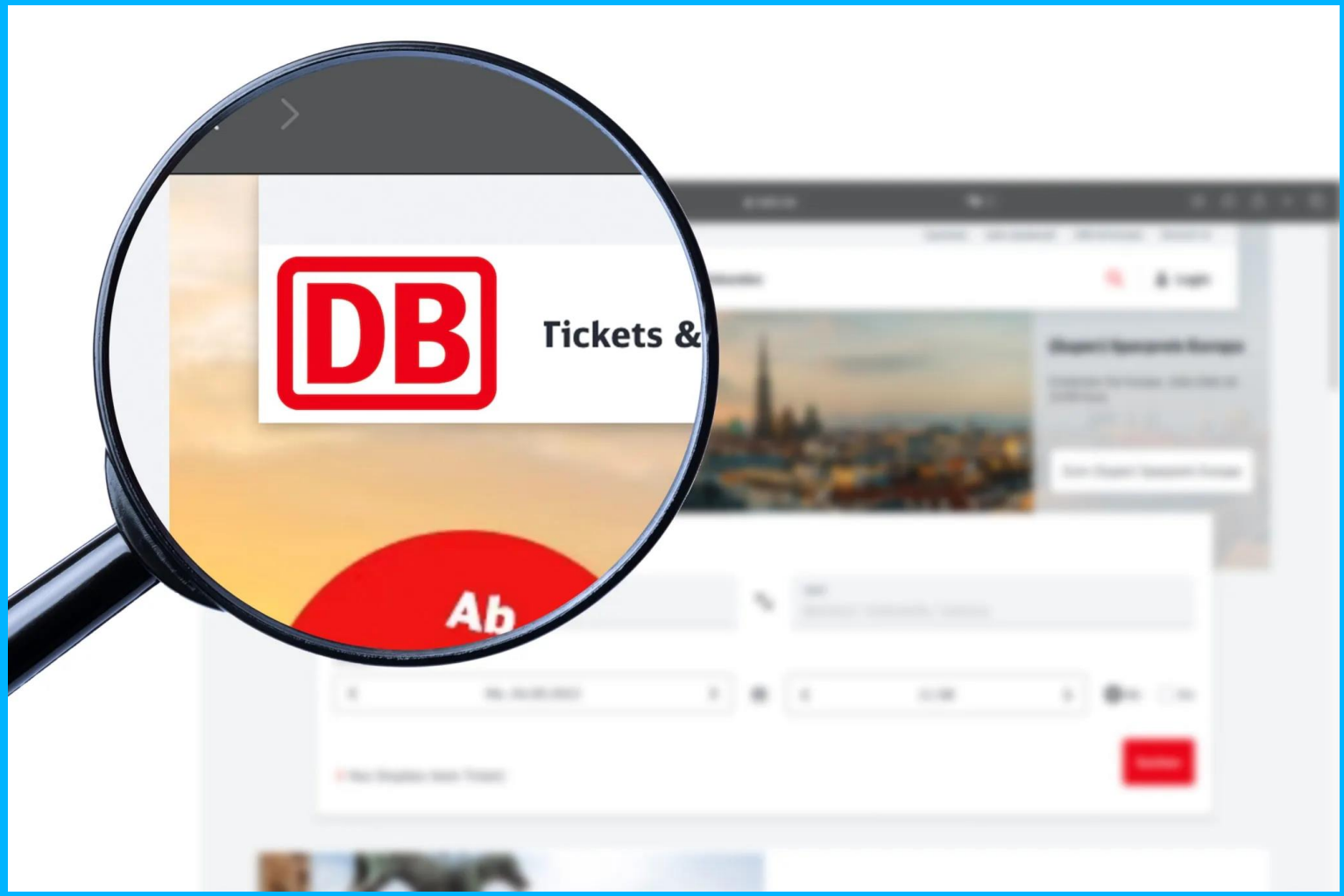
- Projekt – Bahn.de
- Vue 3 – Migrationsumfang
- Problemdarstellung und Lösungsfindung
- JSCodeshift
- Programmierbeispiel
- Fazit und Ausblick

Referenten

Michael Didion
IT-Berater - Conciso GmbH

Melvin McMonagle
Software Entwickler – Deutsche Bahn AG Fernverkehr

Projekt bahn.de



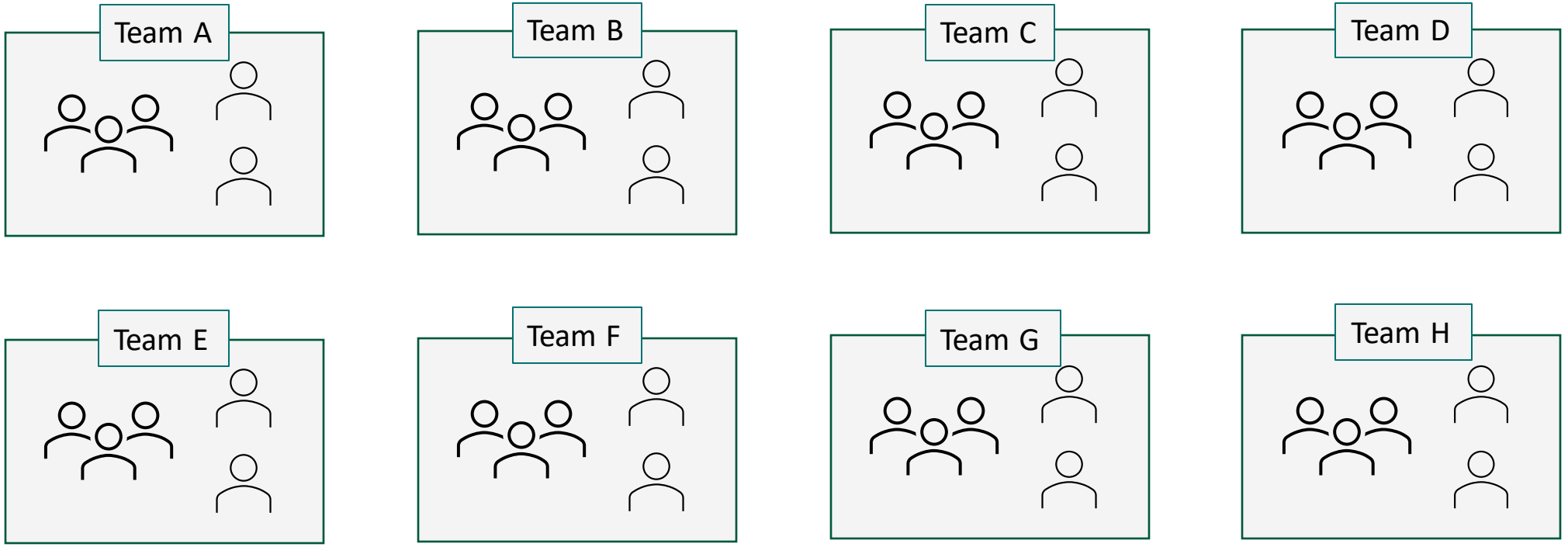
Quellenangabe: Oleksandr – stock.adobe.com

Projektumfang

- 1150 Vue Komponenten
- 4 Module
- Testdriven Development (Unit-Test Coverage ~95%)

- Live: September 2023
- Fortlaufenden Implementierung

Organisation – Scrum Teams

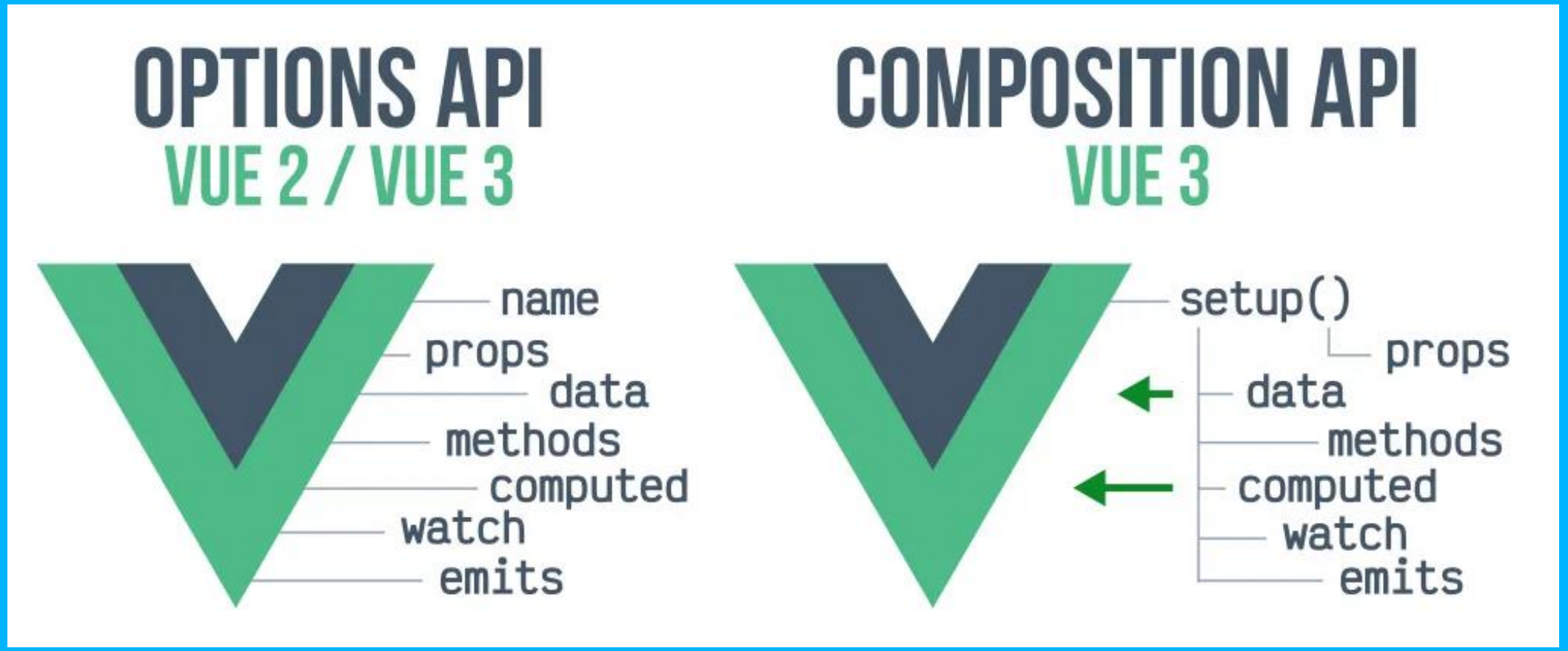


Technologien

- JavaScript
- Typescript
- Vue 2 / 3
- Vee-validate
- Vue-router
- Vue-i18n
- class-validator

- Vite
- Nuxt

Problemdarstellung



Quellenangabe: Haidi Zakelšek - medium.com

Vue 3 – Ein neues Framework?

```vue
export default {
    name: 'ExampleComponent',
    data() {
        return {
            title: 'Hello World'
        }
    },
    props: {
        externalTitle: string
    },
    computed: {
        computedTitle() {
            return this.externalTitle || this.title
        }
    },
    watch: {
        title(newValue, oldValue): {
            console.log(`Titel geändert von ${oldValue} zu ${newValue}`)
        }
    }
}
```

```vue
const title = ref('Hello World');

const props = defineProps({
    externalTitle: String
});

const computedTitle = computed(() => props.externalTitle || title.value);

watch(title, (newValue, oldValue) => {
    console.log(`Titel geändert von ${oldValue} zu ${newValue}`)
});
```

3rd Party Dependencies

- Vue-class-component
  - Deprecated

```
@Component
export default class PropAndWatchExample extends Vue {
    @Prop({ default: 'Startwert' }) private text!: string

    private message: string = 'Nachricht';

    @Watch('text')
    private onTextChange(newValue: string, oldValue: string): void {
        this.message = `Nachricht geändert von ${oldValue} zu ${newValue}`;
    }
}
```

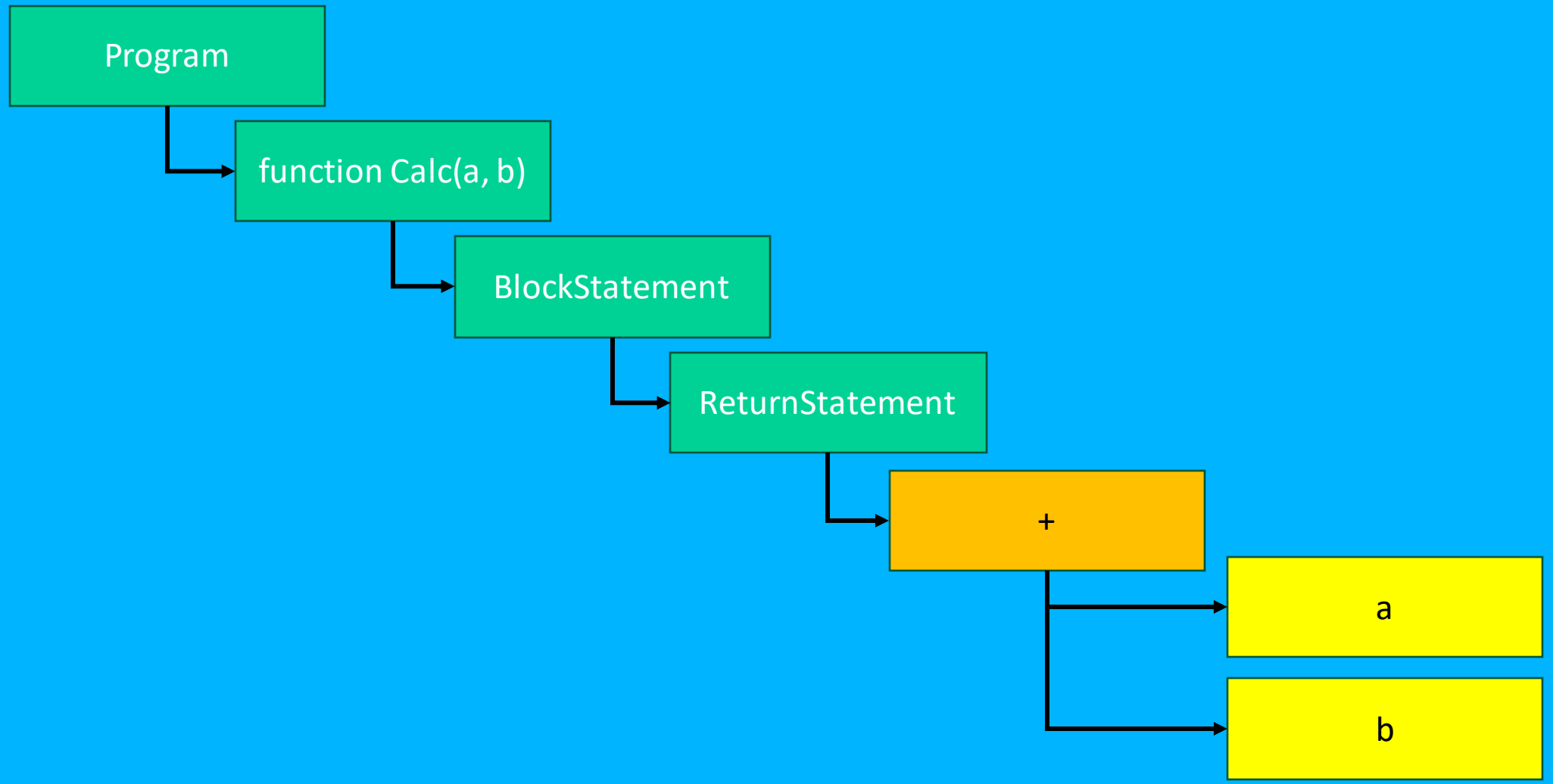
- Vee-validate
  - Viele Changes

Probleme im Deutsche Bahn Kontext

- Fortlaufende Implementierung
- Viele Teams arbeiten parallel an der Umstellung
- EOL 31.12.2023
- Livegang 09.2023

OpenSource

JSCodeshift



JSCodeshift

Toolkit zum Ausführen von **Codemods** auf JavaScript und TypeScript Dateien.

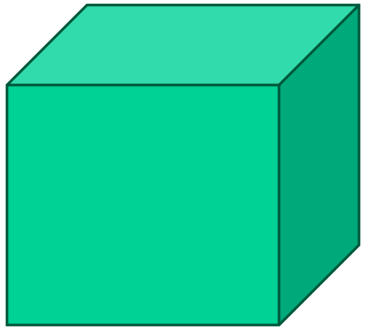
Codemods

- Suchen & Ersetzen von Code zu Code
- Verwendung eines **Abstract Syntax Trees**

Abstract Syntax Tree

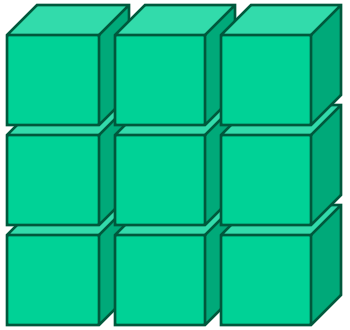
Datenstruktur zur Darstellung des Quellcodes

Node



Node-Path

Collection



Abstrakt Syntax Tree

```
export default {

    methods: {

        greet(): string {
            return 'Hello';
        }

        greetAsync(): Promise<string> {
            return await Promise.resolve('Hello');
        }

    }

}
```

Abstrakt Syntax Tree

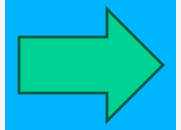
```
export default {

    methods: {
        greet(): string {
            return 'Hello';
        }

        greetAsync(): Promise<string> {
            return await Promise.resolve('Hello');
        }
    }

}
```

Collection<NodePath>

NodePath<FunctionDeclaration>

Programmbeispiel - Codemod

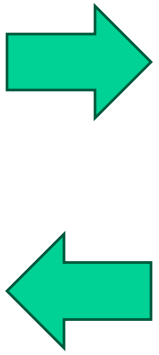
```
export default {

    methods: {

        greet(): string {
            return 'Hello';
        }

        greetAsync(): Promise<string> {
            return await Promise.resolve('Hello');
        }

    }

}
```



```
function greet(): string {
    return 'Hello';
}

async function greetAsync(): Promise<string> {
    return await Promise.resolve('Hello');
}
```

Start & Output

```
export default {

    methods: {

        greet(): string {
            return 'Hello';
        }

        greetAsync(): Promise<string> {
            return await Promise.resolve('Hello');
        }

    }

}
```



```
module.exports = function(fileInfo, api, options)
{
    const {j} = api;
    const root = j(fileInfo.source);

    return root.source();
};

module.exports.parser = 'typescript';
```

Suchen & Durchlaufen

```
export default {

    methods: {

        greet(): string {
            return 'Hello';
        }

        greetAsync(): Promise<string> {
            return await Promise.resolve('Hello');
        }

    }

}
```

```
const collection = root.find(j.ObjectProperty, {
    key: { name: 'methods' }
});

collection.forEach(nodePath => {
    const methods = nodePath.node.properties;

    methods.forEach(() => { /* ... */ });
});
```

Suchen & Durchlaufen

```
export default {
    methods: {

        greet(): string {
            return 'Hello';
        }

        greetAsync(): Promise<string> {
            return await Promise.resolve('Hello');
        }

    }
}
```

```
const collection = root.find(j.ObjectProperty, {
    key: { name: 'methods' }
});

collection.forEach(nodePath => {
    const methods = nodePath.node.properties;

    methods.forEach(() => { /* ... */ });
});
```

Suchen & Durchlaufen

```
export default {

    methods: {

        greet(): string {
            return 'Hello';
        }

        greetAsync(): Promise<string> {
            return await Promise.resolve('Hello');
        }

    }

}
```

```
const collection = root.find(j.ObjectProperty, {
    key: { name: 'methods' }
});

collection.forEach(nodePath => {
    const methods = nodePath.node.properties;

    methods.forEach(() => { /* ... */ });
});
```

Generierung neuer Nodes

```
export default {

    methods: {

        greet(): string {
            return 'Hello';
        }

        greetAsync(): Promise<string> {
            return await Promise.resolve('Hello');
        }

    }

}
```

```
methods.forEach(method => {
    const declaration = j.functionDeclaration(
        method.key.name,
        method.params,
        method.body
    );

    declaration.async = method.async;
    declaration.comments = method.comments;
    declaration.returnType = method.returnType;

    root.insertAfter(declaration);
});
```

Generierung neuer Nodes

```
export default {

    methods: {

        greet(): string {
            return 'Hello';
        }

        greetAsync(): Promise<string> {
            return await Promise.resolve('Hello');
        }

    }

}
```

```
methods.forEach(method => {
    const declaration = j.functionDeclaration(
        method.key.name,
        method.params,
        method.body
    );

    declaration.async = method.async;
    declaration.comments = method.comments;
    declaration.returnType = method.returnType;

    root.insertAfter(declaration);
});
```

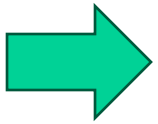
Generierung neuer Nodes

```
async function greetAsync(): Promise<string> {
    return await Promise.resolve('Hello');
}
```

```
methods.forEach(method => {
    const declaration = j.functionDeclaration(
        method.key.name,
        method.params,
        method.body
    );

    declaration.async = method.async;
    declaration.comments = method.comments;
    declaration.returnType = method.returnType;

    root.insertAfter(declaration);
});
```

Ergebnis

```
export default {

    methods: {

        greet(): string {
            return 'Hello';
        }

        greetAsync(): Promise<string> {
            return await Promise.resolve('Hello');
        }

    }

}
```



```
function greet(): string {
    return 'Hello';
}

async function greetAsync(): Promise<string> {
    return await Promise.resolve('Hello');
}
```

War es das Wert?

- Zeitersparnis
- Einheitlichkeit
- Code Cleanup
- Gute teamübergreifende Anwendbarkeit
- OpenSource

Probleme & Hindernisse

- ~ 80% der Migrationen wurden mit Codemods umgesetzt
  - Manuelle Anpassungen notwendig
- Anpassung der Code Konventionen
- Viele unterschiedliche Schreibweisen in Vue
- Umstellung der Module
  - Parallele Nutzung von Vue2 und Vue3

Danke für die Aufmerksamkeit!

Fragen?