Vue.js テスト開発チートシート 🚀

📚 基本セットアップ

必要なインポート

```javascript
import { mount, createLocalVue, shallowMount } from '@vue/test-utils'
import Component from '@/components/Component.vue'
import flushPromises from 'flush-promises'
import Router from 'vue-router'
import Vuex from 'vuex'
```

ローカルVue環境の作成

```javascript
const localVue = createLocalVue()
localVue.use(Router)     // ルーター使用
localVue.use(Vuex)       // Vuex使用
```

🔧 コンポーネントマウント

基本マウント

```javascript
const wrapper = mount(Component, {
  localVue,
  propsData: { prop1: 'value' },     // props渡し
  data() { return { data1: 'value' } } // data設定
})
```

浅いマウント（子コンポーネントをスタブ化）

```javascript
const wrapper = shallowMount(Component, {
  localVue,
  stubs: ['ChildComponent']
})
```

完全設定

```javascript
const wrapper = mount(Component, {
  localVue,
  store,                              // Vuexストア
  router: new Router(),               // ルーター
  mocks: {                            // モック
    $cookies: mockCookies,
    $t: (key) => key                  // i18n
  },
  stubs: ['ExternalComponent'],       // スタブ化
  propsData: { userId: 123 },         // props
  attachTo: document.body             // DOM接続
})
```

🏪 Vuexストア設定

基本ストア作成

```javascript
let store
beforeEach(() => {
  store = new Vuex.Store({
    state: { user: null },
    mutations: {
      SET_USER: (state, user) => { state.user = user }
    },
    actions: {
      fetchUser: jest.fn().mockResolvedValue({ id: 1 })
    },
    getters: {
      isLoggedIn: state => !!state.user
    }
  })
})
```

モジュール分割ストア

```javascript
store = new Vuex.Store({
  modules: {
    auth: {
      namespaced: true,
      state: { token: null },
      actions: {
        login: jest.fn().mockResolvedValue()
      },
      mutations: {
        SET_TOKEN: (state, token) => { state.token = token }
      },
      getters: {
        isAuthenticated: state => !!state.token
      }
    }
  }
})
```

🔄 テストライフサイクル

セットアップ・クリーンアップ

```javascript
describe('Component', () => {
  let wrapper
  let store

  beforeEach(() => {
    // 各テスト前の準備
    store = new Vuex.Store({...})
    wrapper = mount(Component, { store, localVue })
  })

  afterEach(() => {
    // 各テスト後のクリーンアップ
    wrapper.destroy()
    wrapper.vm.$router.push({ query: {} })
  })

  beforeAll(() => {
    // 全テスト前の準備（1回のみ）
  })

  afterAll(() => {
    // 全テスト後のクリーンアップ（1回のみ）
  })
})
```

🎯 DOM要素操作

要素の検索

```javascript
// CSS セレクター
wrapper.find('.button')              // クラス
wrapper.find('#submit')              // ID
wrapper.find('button[type="submit"]') // 属性

// コンポーネント検索
wrapper.findComponent(ChildComponent)
wrapper.findAllComponents(ChildComponent)

// データ属性
wrapper.find('[data-testid="submit-btn"]')
```

要素の存在確認

```javascript
expect(wrapper.find('.error').exists()).toBe(true)
expect(wrapper.find('.success').exists()).toBe(false)
expect(wrapper.findComponent(Modal).exists()).toBe(true)
```

テキスト・HTML確認

```javascript
expect(wrapper.text()).toContain('Hello')
expect(wrapper.html()).toContain('<span>Hello</span>')
expect(wrapper.find('h1').text()).toBe('Title')
```

属性・プロパティ確認

```javascript
expect(wrapper.find('input').attributes('type')).toBe('email')
expect(wrapper.find('button').attributes('disabled')).toBe('disabled')
expect(wrapper.props('visible')).toBe(true)
```

🖱️ イベント・ユーザー操作

クリックイベント

```javascript
await wrapper.find('.button').trigger('click')
await wrapper.find('button').trigger('click')
```

フォーム入力

```javascript
// 入力値設定
await wrapper.find('input').setValue('test@example.com')
await wrapper.find('textarea').setValue('message')
await wrapper.find('select').setValue('option1')

// チェックボックス
await wrapper.find('input[type="checkbox"]').setChecked(true)
```

カスタムイベント

```javascript
await wrapper.find('.component').trigger('custom-event', { data: 'value' })
```

キーボードイベント

```javascript
await wrapper.find('input').trigger('keydown.enter')
await wrapper.find('input').trigger('keyup.escape')
```

⏰ 非同期処理

Promise待機

```javascript
// 非同期処理完了待機
await flushPromises()

// DOM更新待機
await wrapper.vm.$nextTick()

// 組み合わせ
wrapper.find('.button').trigger('click')
await flushPromises()
await wrapper.vm.$nextTick()
```

setTimeout/setInterval

```javascript
// タイマー制御
jest.useFakeTimers()
wrapper.find('.button').trigger('click')
jest.runAllTimers()  // 全タイマー実行
jest.useRealTimers() // 実タイマーに戻す
```

🎭 モック・スパイ

API モック

```javascript
import MockAdapter from 'axios-mock-adapter'
const mockAxios = new MockAdapter(api)

// 成功レスポンス
mockAxios.onGet('/users').reply(200, { users: [] })
mockAxios.onPost('/login').reply(200, { token: 'abc123' })

// エラーレスポンス
mockAxios.onGet('/error').reply(500)
```

関数モック

```javascript
// Jest モック
const mockFunction = jest.fn()
const mockFunction = jest.fn().mockReturnValue('result')
const mockFunction = jest.fn().mockResolvedValue('async result')
const mockFunction = jest.fn().mockRejectedValue(new Error('fail'))

// 呼び出し確認
expect(mockFunction).toHaveBeenCalled()
expect(mockFunction).toHaveBeenCalledWith('arg1', 'arg2')
expect(mockFunction).toHaveBeenCalledTimes(2)
```

メソッドスパイ

```javascript
// Vue インスタンスメソッド
const spy = jest.spyOn(wrapper.vm, 'methodName')
wrapper.find('.button').trigger('click')
expect(spy).toHaveBeenCalled()
```

🧪 よくあるテストパターン

コンポーネント初期化テスト

```javascript
it('正しく初期化される', () => {
  expect(wrapper.exists()).toBe(true)
  expect(wrapper.find('.title').text()).toBe('Expected Title')
  expect(wrapper.vm.loading).toBe(false)
})
```

プロパティテスト

```javascript
it('propsを正しく受け取る', () => {
  const wrapper = mount(Component, {
    propsData: { title: 'Test Title' }
  })
  expect(wrapper.find('h1').text()).toBe('Test Title')
})
```

データ変更テスト

```javascript
it('データが正しく更新される', async () => {
  await wrapper.setData({ count: 5 })
  expect(wrapper.vm.count).toBe(5)
  expect(wrapper.find('.count').text()).toBe('5')
})
```

イベント発火テスト

```javascript
it('クリック時にイベントが発火される', async () => {
  await wrapper.find('.button').trigger('click')
  expect(wrapper.emitted('click')).toBeTruthy()
  expect(wrapper.emitted('click')[0]).toEqual(['arg1'])
})
```

ルーティングテスト

```javascript
it('ルートが正しく変更される', async () => {
  await wrapper.find('.nav-link').trigger('click')
  expect(wrapper.vm.$route.path).toBe('/expected-path')
})
```

ストアアクションテスト

```javascript
it('ストアアクションが呼ばれる', async () => {
  await wrapper.find('.button').trigger('click')
  expect(store._actions['auth/login'][0]).toHaveBeenCalled()
})
```

条件分岐テスト

```javascript
it('条件によって表示が変わる', async () => {
  // 初期状態
  expect(wrapper.find('.error').exists()).toBe(false)

  // エラー状態に変更
  await wrapper.setData({ hasError: true })
  expect(wrapper.find('.error').exists()).toBe(true)
})
```

✅ よく使うマッチャー

基本マッチャー

```javascript
// 等価性
expect(value).toBe(4)                    // 厳密等価
expect(value).toEqual({ name: 'test' })  // 深い等価
expect(value).not.toBe(null)             // 否定

// 存在
expect(value).toBeDefined()              // 定義済み
expect(value).toBeUndefined()            // 未定義
expect(value).toBeNull()                 // null
expect(value).toBeTruthy()               // truthy
expect(value).toBeFalsy()                // falsy

// 文字列
expect(text).toContain('hello')          // 部分一致
expect(text).toMatch(/pattern/)          // 正規表現

// 配列
expect(array).toContain(item)            // 要素含有
expect(array).toHaveLength(3)            // 長さ

// 例外
expect(() => { throw new Error() }).toThrow()
```

Vue固有マッチャー

```javascript
// DOM
expect(wrapper.exists()).toBe(true)
expect(wrapper.isVisible()).toBe(true)
expect(wrapper.text()).toBe('Hello')
expect(wrapper.html()).toContain('<div>')

// イベント
expect(wrapper.emitted()).toHaveProperty('click')
expect(wrapper.emitted('click')).toHaveLength(1)

// Props/Data
expect(wrapper.props('title')).toBe('Test')
expect(wrapper.vm.data).toBe('value')
```

🚨 トラブルシューティング

よくあるエラーと解決法

"Cannot read property of undefined"

```javascript
// ❌ 問題
expect(wrapper.vm.user.name).toBe('John')

// ✅ 解決
expect(wrapper.vm.user?.name).toBe('John')
// または
if (wrapper.vm.user) {
  expect(wrapper.vm.user.name).toBe('John')
}
```

"wrapper.vm.$nextTick is not a function"

```javascript
// ❌ 問題
await wrapper.vm.$nextTick

// ✅ 解決
await wrapper.vm.$nextTick()
```

非同期処理が完了しない

```javascript
// ✅ 解決パターン
await flushPromises()        // Promise待機
await wrapper.vm.$nextTick() // DOM更新待機
jest.runAllTimers()          // タイマー実行
```

💡 ベストプラクティス

テスト設計

```
javascript
```

✅ DO
- 1つのテストで1つの機能をテスト
- 分かりやすいテスト名を付ける
- beforeEach/afterEachでクリーンアップ
- data-testid を使用してDOM要素を特定

❌ DON'T
- 複数の機能を1つのテストに詰め込む
- 他のテストに依存するテストを書く
- マジックナンバーを使用
- 脆い CSS セレクターに依存

パフォーマンス

```
javascript
```

```javascript
// shallowMount を積極的に使用
const wrapper = shallowMount(Component)

// 必要な時だけ mount を使用
const wrapper = mount(Component, {
  stubs: ['HeavyComponent']  // 重いコンポーネントはスタブ化
})
```

🔗 便利なリンク

- Vue Test Utils 公式ドキュメント
- Jest 公式ドキュメント
- Vue.js テストガイド

📝 テンプレート

基本テストファイルテンプレート

```javascript
import { mount, createLocalVue } from '@vue/test-utils'
import Component from '@/components/Component.vue'
import Vuex from 'vuex'

const localVue = createLocalVue()
localVue.use(Vuex)

describe('Component.vue', () => {
  let wrapper
  let store

  beforeEach(() => {
    store = new Vuex.Store({
      // ストア設定
    })

    wrapper = mount(Component, {
      localVue,
      store,
      propsData: {
        // props設定
      }
    })
  })

  afterEach(() => {
    wrapper.destroy()
  })

  it('正しく初期化される', () => {
    expect(wrapper.exists()).toBe(true)
  })

  it('ユーザー操作に正しく反応する', async () => {
    await wrapper.find('.button').trigger('click')
    expect(wrapper.emitted('click')).toBeTruthy()
  })
})
```

このチートシートを手元に置いて、効率的なVue.jsテスト開発を進めましょう！🚀