

Тесты это просто

Буглов Вячеслав Андреевич
Lead Frontend



Пара слов про опыт

- В frontend разработке 5 лет
- Управляю разработкой фронтенд части приложений, планирую задачи и архитектуру приложения
- Основной опыт: Инструменты для анализа данных и упрощения принятия решений



Структура доклада

Доклад в первую очередь предназначен для фронтенд разработчиков.

В первой части доклада я расскажу о преимуществах TDD подхода, во второй части доклада будут приведены практические примеры для тестирования vue приложения.

- Теория - проблемы и решения
- Практика



Разработаем приложение

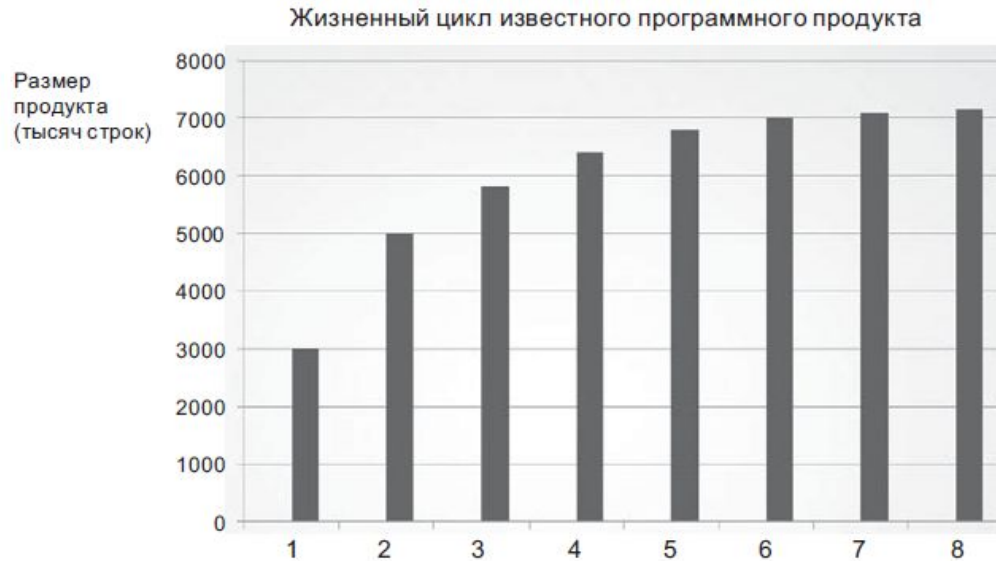


Рис. 1.2. Продуктивность за тот же период

Проблемы

- Необходимо прокликать миллион страниц и компонентов после каждого изменения
- Приложение может сбилдиться, но в маунте какого компонента есть ошибка которая может сломать страницу/компонент
- Бизнес требует миллион изменений в секунду
- Бэкендеры тоже любят вносить изменения вплоть до новой версии api



Почему мы не пишем тесты?

why not if yes can

Мысль: Мы не пишем тесты не потому что мы не успеваем. Мы не успеваем потому что мы не пишем тесты.

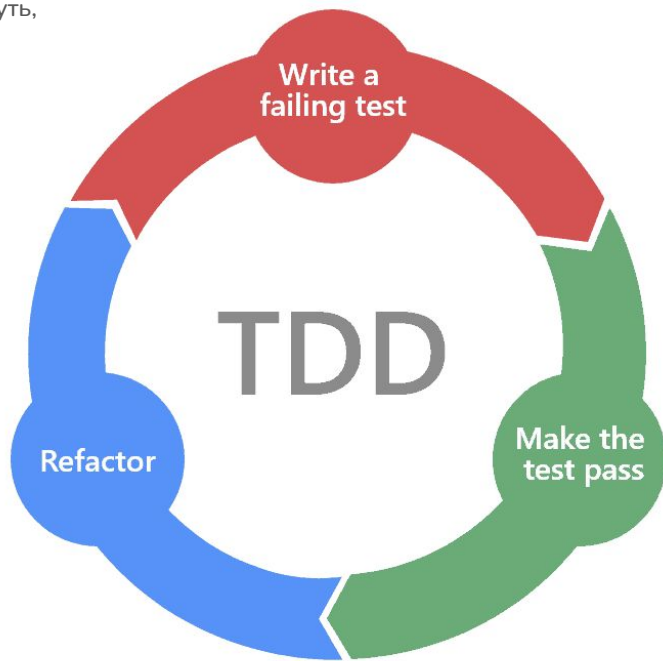
Единственная надежная метрика качества кода:
количество «чертей» в минуту



С любезного разрешения Тома Холверда (Thom Holwerda)
(http://www.osnews.com/story/19266/WTFs_m)

TDD

мне понадобилась неделя чтобы привыкнуть,
сначала непривычно писать с тестами,
но через какое - то время писать без
тестов становится неприятно



Тесты ускоряют разработку

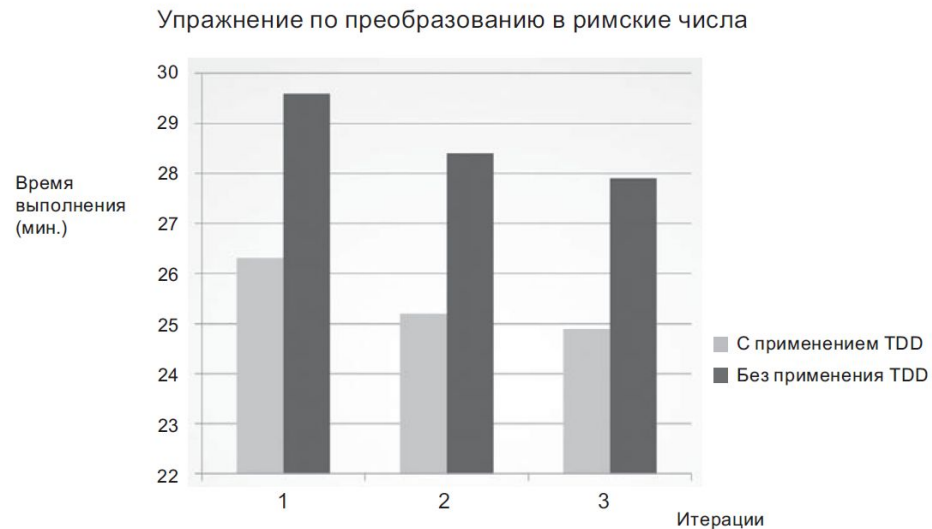


Рис. 1.6. Время на выполнение итерации с использованием и без использования методики TDD

TDD ускоряет разработку

Другие ресурсы

статья

Test-Driven Development (TDD) in Small Software Development Teams:
Advantages and Challenges



Интерактив: SOLID

- Кто руководствуется принципами SOLID при разработке?
- Пишите ли вы тесты?

SOLID: Р. К. Мартин. Чистая архитектура страница 75.

ЧИСТАЯ АРХИТЕКТУРА

ИСКУССТВО РАЗРАБОТКИ
ПРОГРАММНОГО
ОБЕСПЕЧЕНИЯ



РОБЕРТ МАРТИН 

Почему TDD

- Повышение SOLIDности приложения
- Рассуждение о поставленной задаче
- Тесты после кода заставляют переписывать код так его становится сложно тестировать

В результате размышления уменьшается лишнее переписывание кода в результате непонятой/неполной задачи. После написания тестов можно обсудить с руководителем ожидания результатов выполнения задачи исходя из написанных тестов.



Какие тесты нужны

- Архитектурные тесты - 1-2 позитивных теста для компонента/функции.
0-2 негативных кейса.
- Полный цикл тестов уже напишут тестировщики
- Присутствие сайд эффектов внутри функций/компонентов заставляет нас писать дополнительный код для инициализации функции/компонента.
- Наш мозг ленивый и для того чтобы снизить количество лишнего кода мы начинаем писать код без сайд эффектов или сложных ненужных зависимостей.



Библиотеки для тестирования



Vue Test Utils Test Utils for Vue.js 3

The official testing suite utils for Vue.js 3

[Get Started →](#)

Далее будут приведены пример написания тестов на vitest + vue test utils.

P.S. Но еще я покажу пример компонента на чистом js, там vue test utils не нужны

факт: когда я перевел приложение на ts(новые модули) оставив старый код на vue/js, js тесты отлично продолжили работать. Я дописывал просто новые модули на ts

А может другие библиотеки?



VS



import VS require

У библиотек общий интерфейс(названия функций + методы)

Интерактив: библиотеки



VS



BOOKS-UI

- > .idea
- > node_modules
- > public
- > src
- ▼ tests

- > adapters
- > components
- > helpers
- > mocks
- > services
- > store
- > useCases
- ▼ utils

JS tableTests.js

JS ____eslinttrc.cjs

📄 .env

📄 .env.local

📄 .eslinttrc.cjs

📄 .gitignore

📄 Dockerfile

Папка с тестами в корне проекта

Храним папку с тестами отдельно от src. В папке test повторяем вложенность папок проекта.

Typescript config

Для ts укажите папку tests дабы
кодовый редактор не жаловался

tsconfig.json

```
    },  
    "include": ["src/**/*.ts", "src/**/*.tsx", "src/**/*.vue", "tests/**/*.ts"],  
    "references": [{ "path": "./tsconfig.node.json" }]  
  }  
}
```

Виртуальный DOM

Для тестирования верстки необходимо указать библиотеку виртуального рендера. В примере мы будем использовать happy dom.

vite.config.ts

```
30  fx plugins: [vue()],
31  fx test: {
32  fx   globals:true,
33  fx   environment: 'happy-dom',
34     },|
35   })
36
```

Пишем 3000 тестов
— в наносекунду



Тесты простых компонентов

```
import {test, describe, expect} from 'vitest'
import { mount } from '@vue/test-utils'
import BackgroundImage from '@atoms/BackgroundImage.vue';

describe("tests of BackgroundImage", () => {
  test('mount test of BackgroundImage', async () => {

    const wrapper = mount(BackgroundImage, {
      shallow: true,
      props: {
        imageLink: "https://test.com"
      }
    })

    expect(wrapper.exists()).toBe(true)
  })
})
```

```
<script>
export default {
  data() {
    return {
      author: "Robert C. Martin"
    }
  },
  mounted() {},
  methods: {
    changeAuthor () {
      this.author = "Steve McConnell"
    }
  }
}
</script>

<template>
  <div>
    {{ author }}
  </div>
  <button
    id="test-button"
    @click="changeAuthor"
  >
    click me!
  </button>
</template>
```

Тест работы функций



```
import {test, describe, expect} from 'vitest'
import { mount } from '@vue/test-utils'
import WorldBooks from '@organisms/WorldBooks/WorldBooks.vue';

describe("tests of WorldBooks", () => {
  test('mount test of WorldBooks', async () => {
    const wrapper = mount(WorldBooks, {
      shallow: true,
    })

    expect(wrapper.exists()).toBe(true)
    console.log(wrapper.html());

    expect(wrapper.html()).contains("Martin")
    expect(wrapper.vm.author).toBe("Robert C. Martin")

    expect(wrapper.vm.changeAuthor).toBeTypeOf('function')
    await wrapper.vm.changeAuthor()
    expect(wrapper.vm.author).toBe("Steve McConnell")
    expect(wrapper.html()).contains("McConnell")

    console.log(wrapper.html());

  })
})
```

Тест работы функций



Тест работы функций

RERUN tests/components/3_organisms/WorldBooks.test.js x10

```
stdout | tests/components/3_organisms/WorldBooks.test.js > tests of WorldBooks > mount test of WorldBooks
<div>Robert C. Martin</div>
<button id="test-button"> click me! </button>
<div>Steve McConnell</div>
<button id="test-button"> click me! </button>
```

```
✓ tests/components/3_organisms/WorldBooks.test.js (1)
  ✓ tests of WorldBooks (1)
    ✓ mount test of WorldBooks
```

```
Test Files  1 passed (1)
Tests       1 passed (1)
Start at    10:15:41
Duration    333ms
```

PASS Waiting for file changes...
press h to show help, press q to quit

Интерактив: Options API vs Composition API

OPTIONS API VUE 2 / VUE 3



COMPOSITION API VUE 3



Подключение роутера

```
const mockRoute = {  
  params: {  
    id: 1  
  }  
}  
  
const mockRouter = {  
  push: vi.fn()  
}
```

```
test('create book from menu', async () => {  
  const wrapper = mount(SideMenu, {  
    shallow: true,  
    global: {  
      plugins: [Router],  
      mocks: {  
        $route: mockRoute,  
        $router: mockRouter  
      },  
    },  
  })  
})
```

```
const Router = createRouter({  
  history: routerHistory,  
  routes  
});
```

```
test('create book from menu', async () => {  
  const store = createStore({  
    plugins: [],  
    modules: {  
      books,  
      layout  
    }  
  })  
})
```

```
const wrapper = mount(SideMenu, {  
  shallow: true,  
  global: {  
    mocks: {  
      $store: store,  
    },  
  }  
})
```

Подключение стора



Pinia

```
npm i -D @pinia/testing
```



Pinia

```
import {test, describe, expect} from 'vitest'
import {mount} from '@vue/test-utils'
import HelloWorld from '@components/HelloWorld.vue';
import { createTestingPinia } from '@pinia/testing'

describe('HelloWorld', () : void => {
  test('HelloWorld mounted', () : void => {
    const wrapper : VueWrapper<..., ComponentPublicInstance> = mount(HelloWorld,
      {
        options: {
          global: {
            plugins: [createTestingPinia()]
          }
        }
      }
    )
    expect(wrapper.exists()).toBe(true)
  })
})
```

Интерактив: Pinia vs Vuex



vs



Ошибка router-link

```
wrapper > mount(ItemLink, {
  RERUN tests/components/4_frames/SideMenu.test.js x5
})
stderr | tests/components/4_frames/SideMenu.test.js > tests of MenuItems > mount test of ItemLink
[Vue warn]: Failed to resolve component: router-link
If this is a native custom element, make sure to exclude it from component resolution via compile
rOptions.isCustomElement.
  at <ItemLink ref="VTU_COMPONENT" >
  at <VTUROOT>

stderr | tests/components/4_frames/SideMenu.test.js > tests of MenuItems > mount test of ItemBook
[Vue warn]: Failed to resolve component: router-link
If this is a native custom element, make sure to exclude it from component resolution via compile
rOptions.isCustomElement.
  at <ItemBook ref="VTU_COMPONENT" >
  at <VTUROOT>
```

и аналогичные ошибки...

Ошибка router-link

```
test('mount test of ItemBook', async () => {  
  const wrapper = mount(ItemBook, {  
    shallow: true,  
    global: {  
      plugins: [Router],  
      mocks: {  
        $route: mockRoute,  
        $router: mockRouter  
      }  
    }  
  })  
})
```

```
books-ui > src > JS main.js
1  import 'tippy.js/dist/tippy.css'
2  import './styles/main.css'
3
4  import { createApp } from 'vue'
5  import App from './App.vue'
6  import ramdaVue from './ramda-vue.js';
7  import { store } from '@store'
8  import VueTippy from 'vue-tippy'
9  import Router from "@router"
10
11  import moment from "moment/dist/moment"
12  import ru from "moment/dist/locale/ru"
13
14  moment.locale('ru', ru)
15
16  createApp(App)
17    .use(Router)
18    .use(ramdaVue)
19    .use(store)
20    .use(VueTippy)
21    .mount('#app')
22
```

Инициализация плагинов

Созданные через vue test utils компоненты не знают про подключенные плагины на уровне приложения и их нужно инициализировать



Тест не подсказывает где ошибка.

```
RERUN tests/components/5_pages/Book.test.js
stderr | Store.dispatch (F:\Projects\zeitment\books-ui\node_modules\vuex\dist\vuex.esm-bundler.js:1038:15)
[vuex] unknown action type: pages/saveCurrentPage
✓ tests/components/5_pages/Book.test.js (1)
  ✓ tests of BookPage (1)
    ✓ mount test of BookPage
Test Files 1 passed (1)
Tests      1 passed (1)
Start at   22:16:14
Duration   766ms
PASS Waiting for file changes...
press h to show help, press q to quit
```

Тест не подсказывает где ошибка.

```
<script>
export default {
  mounted() {

    const type = this.pageConfig.type
    const sectionId = this.pageConfig.sectionId

    if (!bookId) {
      | this.$router.push('/')
    }

    const serviceOfBooks = new ServiceOfBooks(apiOfBooks, store)
    const serviceOfChapters = new ServiceOfChapters(apiOfChapters, store)
    const serviceOfPages = new ServiceOfPages(apiOfPages, store)
    const layoutService = new ServiceOfLayout(store)

    const bookManager = new BookManager(serviceOfBooks, serviceOfChapters,
    serviceOfPages, layoutService)

    bookManager.fetchBookWithPage(bookId, type, sectionId)
```

Тест не подсказывает где ошибка.

```
import BookPage from '@pages/book/book.vue';
import {createStore} from "vuex";
import {store as books} from "@store/modules/books/index.js";
// import {store as pages} from "@store/modules/pages/index.js";
import axios from "axios";
import {apiBookResponse} from "@mocks/books.js";
import {apiTableOfContentResponse} from "@mocks/tableOfContent.js";

vi.mock('axios')

describe("tests of BookPage", () => {
  const store = createStore({
    plugins: [],
    modules: {
      books,
      // pages
    },
  },
})
```

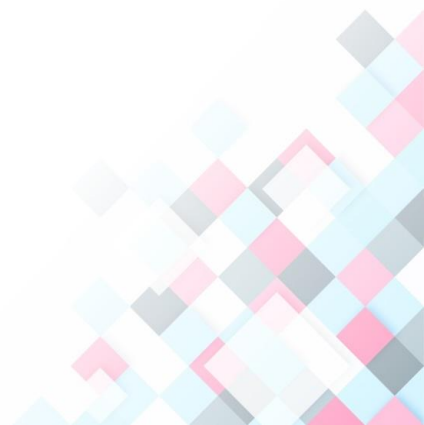
```
describe("bookToMenuItem", () => {
  test('convert book to menu item', () => {
    const menuItem = bookToMenuItem(bookMock)
    const expectedMenuItem = {
      "title": "Тестовая книга",
      "icon": "book-marked-line",
      "link": `/book/${bookMock.id}`,
      "type": "book",
      "name": `${bookMock.id}`
    }

    expect(menuItem).toEqual(expectedMenuItem)
  })
})

describe("booksListToMenuList", () => {
  test('convert books list to menu list', () => {
    const menuList = booksListToMenuList([bookMock])
    const expectedMenuList = [
      {
        "title": "Тестовая книга",
        "icon": "book-marked-line",
        "link": `/book/${bookMock.id}`,
        "type": "book",
        "name": `${bookMock.id}`
      }
    ]

    expect(menuList).toEqual(expectedMenuList)
  })
})
```

Unit тестирование функции



```
import {describe, expect} from 'vitest'  
import sum from "@helpers/sum"  
import tableTests from "../utils/tableTests"
```

```
describe("table test sum function", () => {  
  const testsList = [  
    {  
      title: "golden case: 1+2",  
      expected: 3,  
      params: [1,2],  
      mod: "toBe",  
    },  
    {  
      title: "golden case: 5+2",  
      expected: 7,  
      params: [5,2],  
      mod: "toBe",  
    },  
    {  
      title: "bad case: 5+2",  
      params: [5,2],  
      callback: (value) => expect(value).not.toBe(6)  
    }  
  ]  
  
  tableTests(sum, testsList)  
})
```

Табличные тесты



Табличные тесты

```
import {expect, test} from 'vitest'

type expectMod = 'toEqual' | 'toBe' | 'notToEqual' | 'notToBe'

interface IListOfTests { Show usages new *
  title: string,
  expected: unknown,
  params: Array<unknown>,
  expectMod: expectMod
}

const funcTests = (func: Function, listOfTests: Array<IListOfTests>):void => { no usages new *
  listOfTests.forEach(({title :string , expected, params :?[] , expectMod :expectMod }):void => {
    test(title, () :void => {
      switch (expectMod) {
        case "notToBe":
          expect(func(...params)).not.toBe(expected)
          break;
        case "notToEqual":
          expect(func(...params)).not.toEqual(expected)
          break;
        default :
          expect(func(...params))[expectMod](expected)
          break;
      }
    })
  })
}
```


Мока данных

```
adapters > testBooksApi.test.js > ...  
import {test, describe, vi, expect} from 'vitest'  
import axios from 'axios'  
import AdapterOfBooks from "@adapters/AdapterOfBooks/AdapterOfBooks.js"  
import {appBook, apiBooksResponse} from '@mocks/books.js'  
  
vi.mock('axios')  
  
describe("tests AdapterOfBooks for get books", () => {  
  const url = "http://localhost:8000/api/v1/books/"  
  axios.post.mockResolvedValue({data: apiBooksResponse})  
  const apiOfBooks = new AdapterOfBooks(url)  
  
  test("get books list", async () => {  
    const booksData = await apiOfBooks.getBooks()  
    expect(booksData).toEqual([appBook])  
  })  
})
```

Мока данных

```
20 describe("tests AdapterOfBooks with mock implementation", () => {
21   const url = "http://localhost:8000/api/v1/books/"
22   axios.post.mockImplementation(async (args) => {
23     console.log(args);
24     return {data: apiBooksResponse}
25   })
26   const apiOfBooks = new AdapterOfBooks(url)
27
28   test("get books list", async () => {
29     const booksData = await apiOfBooks.getBooks()
30     expect(booksData).toEqual([appBook])
31   })
32 })
33
```


Тестирование сервисного слоя

```
import {ApiOfBooks as ApiOfBooks, appBook} from "@mocks/books.js"
describe('serviceOfBooks', () => {
  const store = createStore({
    modules: {
      books
    }
  })

  beforeEach(async () => {
    await store.dispatch('books/resetStore')
  })

  const ApiOfBooks = new ApiOfBooks('');
  const serviceOfBooks = new ServiceOfBooks(ApiOfBooks, store);
```

```
class ApiOfBooks { no usages Вячеслав Буглов +1 *
  constructor(uri) { no usages Вячеслав Буглов
    this.uri = uri
  }

  async getBooks() : Promise<[{owner: string, variables: an...}> { S
    return [appBook]
  }

  async updateBook(newBook) : Promise<...> { Show usages Вячеслав Буглов
    return {
      ...appBook,
      ...newBook
    }
  }

  async createBook() : Promise<{...}> { Show usages Вячеслав Буглов
    return appBook
  }

  async deleteBookById() : Promise<{...}> { Show usages Вячеслав Буглов
    return appBook
  }
}
```

Тестирование сервисного слоя мока класса api



```
mounted() {},
methods: {
  logFunction(data) {
    this.testData = [
      ...this.testData,
      data
    ]
  },
  testEvent() {
    const eventsAdapter = new AdapterOfEvents(this.url)
    this.testData = []
    return startIntegrationTests(eventsAdapter, this.logFunction)
  },
  testBook() {
    const booksAdapter = new AdapterOfBooks(this.url)
    this.testData = []
    return startIntegrationTests(booksAdapter, this.logFunction)
  },
  testChapter() {
    const chaptersAdapter = new AdapterOfChapters(this.url)
    this.testData = []
    return startIntegrationTests(chaptersAdapter, this.logFunction)
  },
  testPage() {
    const chaptersAdapter = new AdapterOfPages(this.url)
    this.testData = []
    return startIntegrationTests(chaptersAdapter, this.logFunction)
  },
  testParagraph() {
    const chaptersAdapter = new AdapterOfParagraphs(this.url)
    this.testData = []
    return startIntegrationTests(chaptersAdapter, this.logFunction)
  }
}
```

Тесты интеграции

```
testPage() {
  const chaptersApi = new ApiOfPages(this.url)
  this.testData = []
  return startIntegrationTests(chaptersApi, this.
    logFunction)
},
testParagraph() {
  const chaptersApi = new ApiOfParagraphs(this.url)
  this.testData = []
  return startIntegrationTests(chaptersApi, this.
    logFunction)
}
```

```
async integrationTests(logFunction) {  
  logFunction("Список глав")  
  const books = await this.getChapters()  
  logFunction(books)  
  logFunction("Список глав по книге")  
  const chaptersByBook = await this.getChaptersByBookId  
  ("fb5e7d1d-38cd-4831-bae9-07b36080e3e7")  
  console.log(chaptersByBook)  
  logFunction("Создание главы")  
  const newChapter = await this.createChapter({  
    "title": "Chapter 1",  
    "number": 1,  
    "text": "tests",  
    "bookId": "fb5e7d1d-38cd-4831-bae9-07b36080e3e7",  
    "isPublic": false  
  })  
  logFunction(newChapter)  
  logFunction("Получение главы по id(Созданной)")  
  const bookById = await this.getChapterById(newChapter.id)  
  logFunction(bookById)  
  logFunction("Обновление главы")  
  const updatedChapter = await this.updateChapter({...bookById,  
    title: "Обновленная глава"})  
  logFunction(updatedChapter)  
  logFunction("Удаление главы")  
  const deletedChapter = await this.deleteChapterById(bookById.id)  
  logFunction(deletedChapter)  
}
```

Тесты интеграции



Тест компонента на чистом JS

```
⚡ vite.config.js > ...  
1   import { defineConfig } from 'vite'  
2  
3   // https://vitejs.dev/config/  
4   export default defineConfig({  
5     test: {  
6       globals: true,  
7       environment: 'happy-dom',  
8     }  
9   })
```



```
class SpeedComponent {  
  constructor(container, params) {  
    this.container = container  
    this.params = params  
    this.render()  
  }  
  
  render() {  
    this.container.innerHTML = `  
      <div id="world">  
        hello world  
      </div>  
      <div id="forest">  
        green forest  
      </div>  
    `;  
  }  
}
```

Тест компонента на чистом JS



Тест компонента на чистом JS

```
describe('SpeedComponent', () => {  
  
  test('render test', () => {  
    const container = document.createElement('div')  
    const wrapper = new SpeedComponent(container, {})  
  
    console.log(container.innerHTML)  
  
    expect(wrapper.container.innerHTML).not.contains("slot")  
    expect(wrapper.container.innerHTML).contains("hello world")  
    expect(wrapper.container.querySelector('[id=forest]').innerHTML).contains('green')  
  })  
})
```

Тест компонента на чистом JS

```

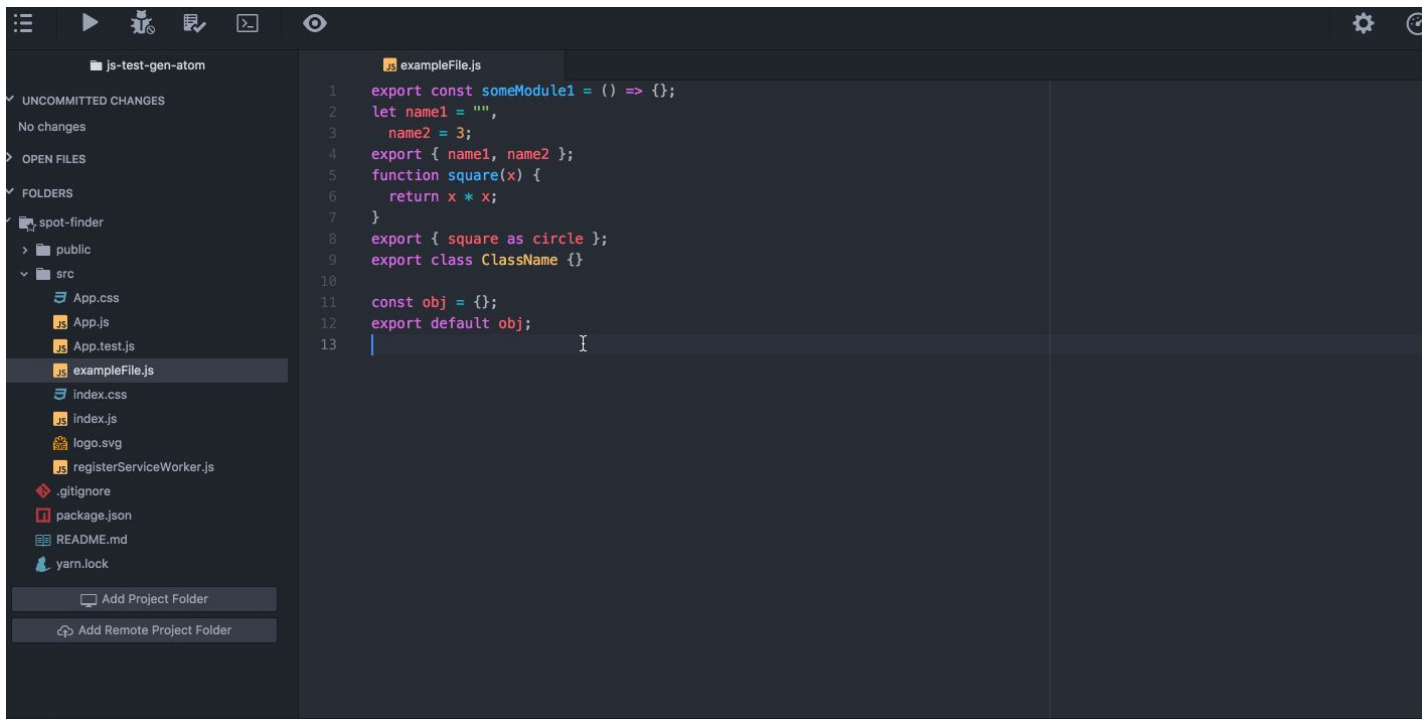
RERUN tests/components/speed.test.js x9
stdout | tests/components/speed.test.js > SpeedComponent > render test
  <div id="world">
    hello world
  </div>
  <div id="forest">
    green forest
  </div>
  ✓ tests/components/speed.test.js (1)
    ✓ SpeedComponent (1)
      ✓ render test

Test Files  1 passed (1)
Tests       1 passed (1)
Start at    23:39:07
Duration    68ms

PASS Waiting for file changes...
press h to show help, press q to quit

```


Генерация тестов



```
1 export const someModule1 = () => {};  
2 let name1 = "",  
3     name2 = 3;  
4 export { name1, name2 };  
5 function square(x) {  
6     return x * x;  
7 }  
8 export { square as circle };  
9 export class ClassName {}  
10  
11 const obj = {};  
12 export default obj;  
13
```

<https://js-test-gen.github.io/>

js-test-gen-atom

UNCOMMITTED CHANGES

No changes

OPEN FILES

FOLDERS

spot-finder

public

src

App.css

App.js

App.test.js

exampleFile.js

index.css

index.js

logo.svg

exampleFile.js

```
1 export const someModule1 = () => {};  
2 let name1 = "",  
3     name2 = 3;  
4 export { name1, name2 };  
5 function square(x) {  
6     return x * x;  
7 }  
8 export { square as circle };  
9 export class ClassName {}  
10  
11 const obj = {};  
12 export default obj;  
13
```

Спасибо за
внимание!



@VBUGLOV

