A

Actor

- An "actor" typically represents an external entity interacting with the system.
- Typically used in block diagrams and sequence diagrams e.g. non-empty-queue-sequence-diagram.png

afterEach

a function in vitest \ jest that invokes its callback after each test function in a test file e.g., in task-scheduler.test.ts

B

beforeEach

a function in vitest \ jest that invokes its callback before each test function in a test file e.g., task-queue-sociable.test.ts

Block diagram

- In software development, a block diagram is a visual representation that illustrates the high-level structure of a system, showing major components and their interactions.
- It simplifies complex systems for design and communication.
- Check e.g. block-diagram.png

Browser extension

a software module that adds functionality or features to a web browser, enhancing its capabilities.

C

Class diagram

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's: classes, their attributes, operations (or methods), and the relationships among objects , check e.g. uml-class-diagram.png

clearAllMocks

function in vitest \ jest to clear all mocks used e.g. in task-queue-isolated.test.ts where it clears the number of spy call before each test

cls

Command to clear the terminal

Code coverage

Code coverage is a software metric for measuring the percentage of code that is executed by test cases during software testing

Ctrl a

In vscode: select all

Ctrl g

In vscode: Go to line number

Ctrl v

In vscode: paste

Ctrl x

In vscode: cut

D

Debug unit test

can be done using vscode->JavaScript debug terminal and breakpoint

Dom testing library

- The package name is @testing-library/dom
- Used in testing library
- Installed using -D e.g. here

Draw.io

- Draw.io is a popular web-based diagramming application that allows users to create a wide range of diagrams and visual representations, for example, block diagram, sequence diagram, class diagram
- Check e.g., here, for files with extension .drawio

E

E2E Test

Testing the entire system as a whole to evaluate its compliance with the specified requirements.

Esm

- Es module
- JavaScript modules format, which is the official standard format to package JavaScript code for reuse

expect

function in vitest \ jest used to expect a value, e.g., here

F

F8

In vscode: go to the next error \ warning

Fake timer

- Mock the timer API
- A common mock package is sinonjs/fake-timers used e.g. in task-scheduler.test.ts

Fake timer.install

- Function of the package sinonjs/fake-timers which is used to create a fake timer used e.g. in task-scheduler.test.ts

fakeClock.uninstall

- Function of the package sinonjs/fake-timers which is used to remove a fake timer used e.g. in task-scheduler.test.ts

final

directory with final code, e.g. here

findByText

- async API of @testing-library/dom used, e.g., in main-ui.test.ts
- Use this API when the dom element does not appear immediately
- CAUTION : You should use this function with await because it returns a promise
- Using findByText, you don't need my utility function pauseMs

Filter test

- npm test test\task-dispatcher.test.ts -> will only run the test in task-dispatcher.test.ts
- npm test test\task-dispatcher.test.ts – -t ‘dispatch result is ok for add’ will run the test with this description in this file

FIFO

- Abbreviation for First In First Out

flushPromises

- Test utility function used to resolve the promises used e.g., in task-scheduler.test.ts
- Used with fake timer and async functions

fn

function in vitest \ jest to replace side effect functionality check e.g. task-queue-isolated.test.ts , task-scheduler.test.ts

G

getAllByRole

- Similar to getByRole by returning a list of dom elements if find few dom elements with the same role

getByRole

- An API of @testing-library/dom used e.g. in main-ui.test.ts
- You can use it e.g. with role ‘heading’ to get dom element that is h1 or h2 or h3 or h4 or h5 or h6

getByText

- An API of @testing-library/dom used e.g., in main-ui.test.ts
- You can use it with text
- It replaces my internal API getButtonInUI for testing library unit tests

getButtonInUI

- An internal utility function defined in test-utils.ts and used for jsdom test e.g. main-ui.test.ts tag 0.6

getEnumKeyValues

- An internal utility function defined in test-utils.ts : given an enum returns its key and value pairs
- This function is used e.g. in main-ui.test.ts

H

I

Integration Test

Testing the interaction between different components or modules of the software to ensure they work together seamlessly.

Isolated test

- Use mock to isolate the unit from side effects and other module interactions e.g. in task-queue-isolated.test.ts
- Called also a solitary test

Istanbul

code converge package used in this course check, e.g., package.json and vite.config.ts

J

Jest

- Unit test framework
- Installed using -D
- Check e.g. final/package.json and final/test-jest

Jest object

- Central object in jest
- Can be used to access functions like spyOn, fn, mock, beforeAll, …

Jest setup

1. pnpm i -D jest @types/jest ts-jest ts-node identity-obj-proxy jest-environment-jsdom
2. Create jest.config.ts, e.g., here 👍
    - testEnvironment - specify the testing environment in which your tests will run, e.g., jsdom
    - transform - used to specify how files should be transformed before they are tested
    - moduleNameMapper - map module names to specific paths or aliases
    - setupFilesAfterEnv - an array of setup files that should be executed after the test framework (Jest) has been set up but before the tests are run

    - testMatch - specify a pattern that Jest will use to match test file paths and determine which files should be included in the test run
3. Add test-jest script in package.json to run jest test
4. Add coverage-jest script in package.json to run jest coverage test
5. Add a setup file e.g. setup-jest.ts, to extend matchers
6. Add to tsconfig.json
    - include - add test-jest here so compilation will also include test-jest directory
    - esModuleInterop: true - this was due to a problem with dayjs

jsdom

- A popular package that emulates the important part of the browser in particular the dom
- Used by dom testing library
- Used in jest.config.ts and vite.config.ts to define client-side unit testing
- Installed using -D e.g. here

K

L

LocalStoragePersist

A class that implements IPersistStorage to be used in a web client . It is using local storage internally

Logic unit test

test logic function, i.e., function without side effects e.g., add two numbers, e.g., here

Logic system

The logic system in this course is actually what we have in lib directory. It by nature has no knowledge of the UI

M

Mock

- definition - refers to a simulated or fake object that is created to mimic the behavior of a real object or component within a software system

mock

function in vitest \ jest used to replace a module check e.g. task-queue-isolated.test.ts

Module interaction

When one module calls an API of another module e.g. TaskSceduler use taskDispatcher.dispatch in task-scheduler.ts

N

not

used with matcher function e.g., in task-scheduler.test.ts

npm run build

Compile the typescript files

npm run dev

Run the UI

npm run test-jest

Run the tests using jest

npm test

Run the tests using vitest

O

P

persist

- A module of persistence as part of TaskQueue Implemented in persistence.ts

pnpm

(npm , yarn) , pnpm i , pnpm i -D

Private methods

- In general, not part of the unit test
- e.g. the private method save of TaskQueue does not appear in the test e.g.task-queue-sociable.test.ts

Promise.reject

- create and return a new Promise object that is rejected with a given value
- Used in task-dispatcher.test.ts

Promise.resolve

- create and return a new Promise object that is resolved with a given value
- Used in task-dispatcher.test.ts

Q

Queue

- Well-known data structure that behave as FIFO
- You can insert to the queue tail using enqueue and remove from the queue head using dequeue.

R

React testing library

- The package name is @testing-library/react
- Very popular testing library for react applications
- Installed using -D e.g. here

Refactoring

The process of restructuring and improving the internal structure of existing code without changing its external behavior e.g., changing persistent object to class here class , here object

render

- Api of react testing library which is used to render a component e.g. in main-ui-react.test.tsx
- You can use render of react-dom in jsdom test see e.g. test\main-ui.test.tsx of tag 0.91 but render of RTL is much more popular to test react application

S

Scheduler

- A scheduler in software manages task execution order

screen

- This object is an essential part of the React Testing Library (RTL) API, and it provides a convenient way to query and interact with elements rendered within your React components during testing e.g. in main-ui-react.test.tsx
- The following API are used in main-ui-react.test.tsx using screen
    - screen.getByText
    - screen.getByRole
    - screen.getAllByRole
    - screen.findByText

Sequence diagram

- A sequence diagram is a UML diagram used in software engineering to visualize system interactions between objects or components.
- It shows the chronological order of messages or method calls between objects, helping to model the behavior and flow of a system.
- Check e.g. non-empty-queue-sequence-diagram.png

Setup.ts

- A setup file of vitest e.g. here used e.g. to extend matcher and use toBeInTheDocument

Side effect

- code that is not pure logic, for example, accessing the network using Axios \ fetch
- This can be handled using mock \ spy in a unit test if we want isolation of the unit from external \ side effects

sinonjs/fake-timers

- Common package to mock the javascript timer API
- Documentation is here

Sociable test

- use the module we depend on in the unit test(halfway to integration) e.g. task-queue-sociable.test.ts here we unit test TaskUsing using persist module

spyOn

function in vitest \ jest to spy on side effects see e.g. persistence.test.ts

src

directory in a project, commonly holds source file e.g. here

starter

directory with initial code, e.g., here

Storage.prototype

Use to spy on localStorage in jsdom check e.g. persistence.test.ts. You can not use here localStorage because of jsdom bug

T

.test.

part of a test file name in vitest \ jest e.g., math.test.ts

Task

- In software programming, a task typically refers to a discrete unit of work or an independent job that a program needs to perform.
- Implemented in i-task.ts

TaskDispatcher

- A module of task dispatcher as part of the system Implemented in task-dispatcher.ts

TaskQueue

- A module of task queue as part of task queue manager Implemented in task-queue.ts

TaskScheduler

- A module of task scheduler as part of the system Implemented in task-scheduler.ts

test

- Manual - performed by a person
- automatic - performed by the PC using software e.g. vitest \ jest
- script (package.json)
- directory in a project commonly holds test files. It can be inside the src directory or a standalone

- function in vitest \ jest, used to define a test case
- description in the test function
- expected value **-** the value that we expect to get from a test e.g. 3 for add(1,2)
- actual value - the actual result of a test e.g. 2 if add(1,2) multiply instead of add

test-jest

This has two meaning in this course

- Test directory for jest check system/final/test-jest
- Script to run jest over test-jest directory check system/final/package.json

Testing

Testing is the process of evaluating a software system to ensure it meets the desired requirements and functions correctly

Testing library

- typically refers to a family of JavaScript testing utilities and libraries that promote best practices for writing more effective and maintainable tests for web applications

testing-library/user-event

- A package of testing library that provides a set of utility functions for simulating user interactions with DOM elements in a more realistic and user-focused way.
- Used e.g. in main-ui.test.ts

textContent

Due to a jsdom bug you should use textContent property instead of innerText when using jsdom chck e.g. main-ui.test.ts

tick

- API of sinonjs/fake-timers used to advance the clock, firing callbacks if necessary. Check e.g. task-scheduler.test.ts

toBe

- simple matcher function in vitest \ jest used e.g. here
- Do not use it to compare reference types like object

toBeCalledTimes

- matcher function in vitest \ jest to check how many times a spy was called. check e.g. persistence.test.ts
- Aliased to toHaveBeenCalledTimes

toBeCalledWith

- matcher function in vitest \ jest to check if a spy was called with the correct arguments. check e.g. persistence.test.ts
- Same as toHaveBeenCalledWith

toBeFalsy

matcher function in vitest \ jest to check if a value is falsy, check, e.g., task-queue-sociable.test.ts

toBeInTheDocument

- Extended matcher used by vitest \ jest e.g. in test-jest/main-ui.test.ts
- Originally from testing-library/jest-dom

toBeTruthy

matcher function in vitest \ jest to check if a value is truthy, check, e.g., main-ui.test.ts

toEqual

- matcher function in vitest \ jest used, e.g., here
- Used, e.g., to compare that object have the same structure

toHaveReturnWith

matcher function in vitest \ jest to check the return value check e.g. persistence.test.ts

toStrictEqual

- matcher function in vitest \ jest used, e.g., task-queue-sociable.test.ts (actually toEqual is enough in this case)
- Used, e.g., to compare that objects have the same structure and order is important

toThrowError

- matcher function in vitest \ jest to check if a promise is rejected check e.g. in system\final\test\task-dispatcher.test.ts in tag 0.31
- Alias to toThrow

U

Unit Test

Testing individual units or components of the software to ensure they function correctly in isolation.

Unit

(in software application) - typically function or class or component

userEvent.click

API from testing-library/user-event , used to simulate click event on dom element e.g. in test/main-ui.test.ts or test-jest/main-ui.test.ts

V

Vanilla vite project

- basic or minimal web application project created using the Vite build tool, often without any additional frameworks or libraries.

- It serves as a starting point for developers to build web applications using Vite's efficient development and build features, with the freedom to add their preferred technologies as needed.
- Example is this

vi

- Central object in vitest
- Can be used to access functions like spyOn, fn, mock, beforeAll, …

Vite

a build tool and development server for web applications that focus on fast development and efficient, near-instantaneous builds using native ES modules in JavaScript and typescript.

vite.config.ts

Configuration file for vite and vitest e.g. system/final/vite.config.ts. The following properties are used in this course:
plugins - extend rollup plugin interface
test - test context, e.g., vitest

- exclude - used, e.g., to ignore test-jest directory
- setupFiles - path to setup file, e.g., system/final/test/setup.ts
- environment - e.g., jsdom, node, ..
- coverage - define the coverage tool, e.g., istanbul

vitest

- Unit test framework
- Installed using -D
- Check e.g., final/package.json and final/test

vitest coverage test - setup

- Install @vitest/coverage-istanbul on dev
- Create in package.json a script "coverage": "vitest run --coverage"
- Create vite.config.ts as e.g. with istanbul follows

```
/// <reference types="vitest" />
import { defineConfig } from "vitest/config";

export default defineConfig({
  test: {
    coverage: {
      provider: "istanbul", // or 'v8'
    },
  },
});
```

W

waitFor

- async API of @testing-library/dom used to wait for an element in the dom e.g., to appear e.g. in test-jest/main-ui.test.ts , test/main-ui.test.ts
- CAUTION : You should use this function with await because it returns a promise
- Using waitFor, you don't need my utility function pauseMs

X

Y

Z