

Distribueret Programming

Lektion 17: Test

Testmoduler



Mocha Test Explorer v2.13.0

Holger Benl | 67.562 | ★★★★★ (24)

Run your Mocha tests in the Sidebar of Visual Studio Code

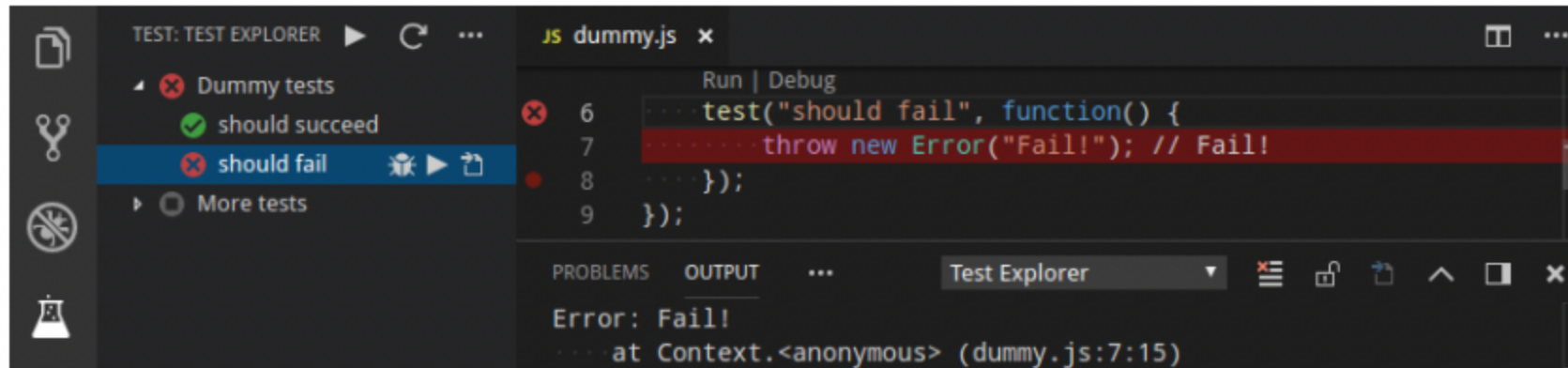
[Disable](#) [Uninstall](#)

This extension is enabled globally.

[Details](#) [Feature Contributions](#) [Dependencies](#) [Runtime Status](#)

Mocha Test Explorer for Visual Studio Code

Run your Mocha tests using the [Test Explorer UI](#).





ES6 Mocha Snippets v0.2.2

Cory Noonan |  63.445 |  (3)

Shortcuts to reduce the amount of boiler plate you need to type when creating a test file us

Disable 

Uninstall 



This extension is enabled globally.

[Details](#)

[Feature Contributions](#)


[Changelog](#)

[Runtime Status](#)

build passing

ES6 Mocha Snippets for Visual Studio Code!

Mocha snippets for Visual Studio Code using ES6 syntax. The focus is to keep the code dry leveraging arrow functions and omitting curly braces by where possible. The snippets use the Mocha function names for ease of memory that way you don't need to learn new names.

```
1 describe('my module', () => {  
2   it('can do th', () => {  
3      throw  
4   });  
5 });
```

Throw Exception 

mocha

Runs Mocha tests

Disable



Uninstall



This extension is enabled globally.

Extension name

Details

Feature Contributions

Runtime Status

Mocha

Runs Mocha tests, all or selected. Then prints the result to an output window.

This extension is inspired by [Node.js Tools for Visual Studio](#).

OUTPUT

Running Mocha with Node.js at C:\Program Files\nodejs\node.exe

No Mocha options are configured. You can set it under File > Preferences > Workspace Settings.

Test file(s):

→ [c:\Users\Compulim\Git\node-openc\test\openc.test.js](#)

→ Convert based on OpenCC tests

✓✓✓✓✓ should convert Hong Kong to Simplified Chinese

✓✓✓✓✓ should convert Simplified Chinese to Hong Kong (218ms)

✓✓✓✓✓ should convert Simplified Chinese to Traditional Chinese (60ms)

✓✓✓✓✓ should convert Simplified Chinese to Taiwan (62ms)

✓✓✓✓✓ should convert Simplified Chinese to Taiwan with phrases (51ms)

✓✓✓✓✓ should convert Traditional Chinese to Simplified Chinese

✓✓✓✓✓ should convert Taiwan to Simplified Chinese

✓✓✓✓✓ should convert Taiwan to Simplified Chinese with phrases

npm pakker

- Mocha

```
$ npm install mocha
```

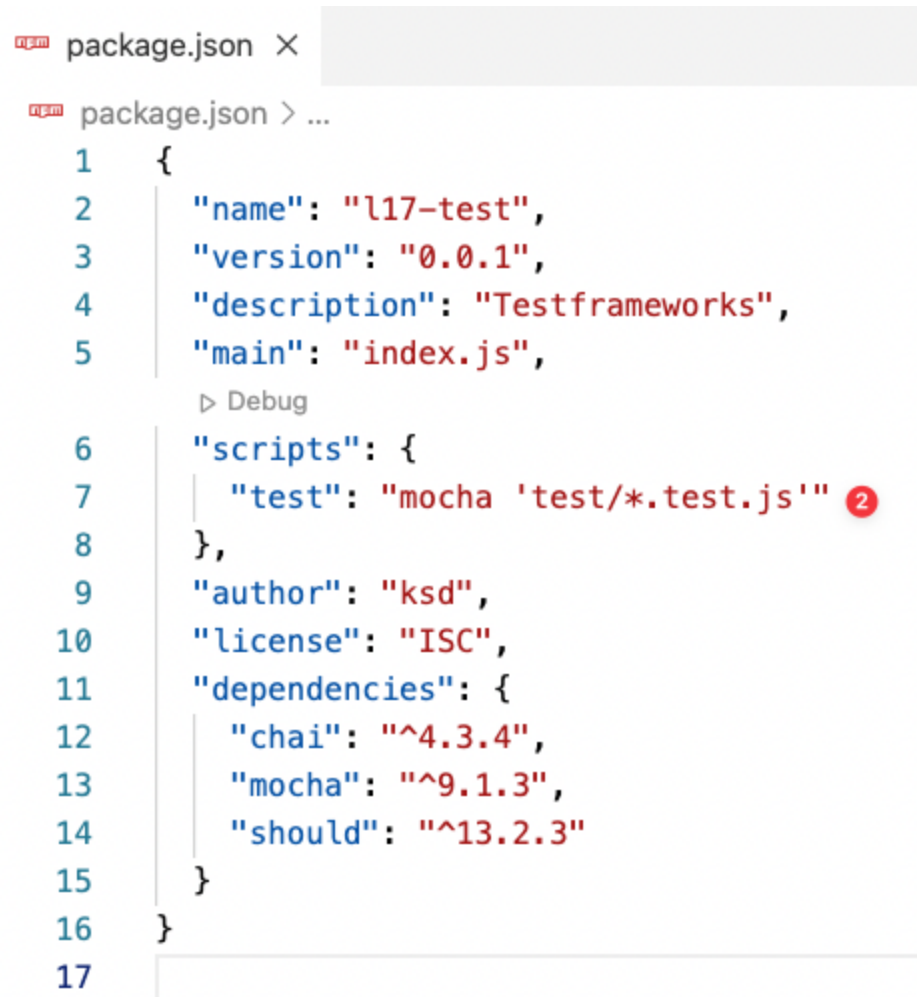
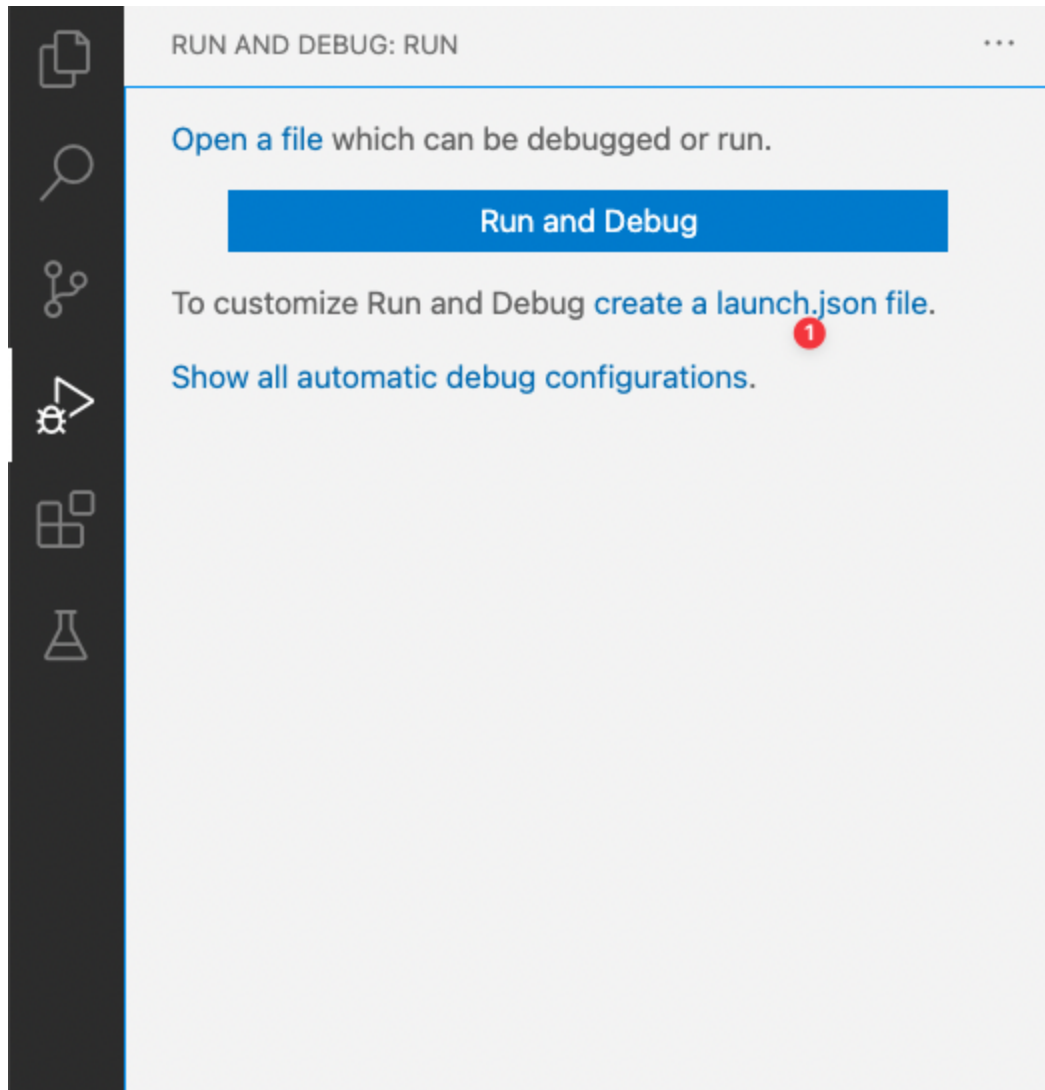
- Chai


```
$ npm install chai
```

Mocha BDD interface


- `describe()` – specificerer en test suite med en beskrivelse
- en callback funktion definerer testene
- `it()` – specificerer en enkelte test med en beskrivelse
- en callback funktion definerer testen
- `before()` – udføres før alle testene i en test suite
- `after()` – udføres efter alle testene i en test suite
- `beforeEach()` – udføres før hver test
- `afterEach()` – udføres efter hver test

Konfigurering af VS-Code







EXPLORER




.vscode




launch.json 1




coverage




test




add.test.js




L17-test.code-workspace




subtract.test.js



index.js



package-lock.json



package.json

> OUTLINE

> TIMELINE

JS index.js

JS add.test.js

JS subtract.test.js

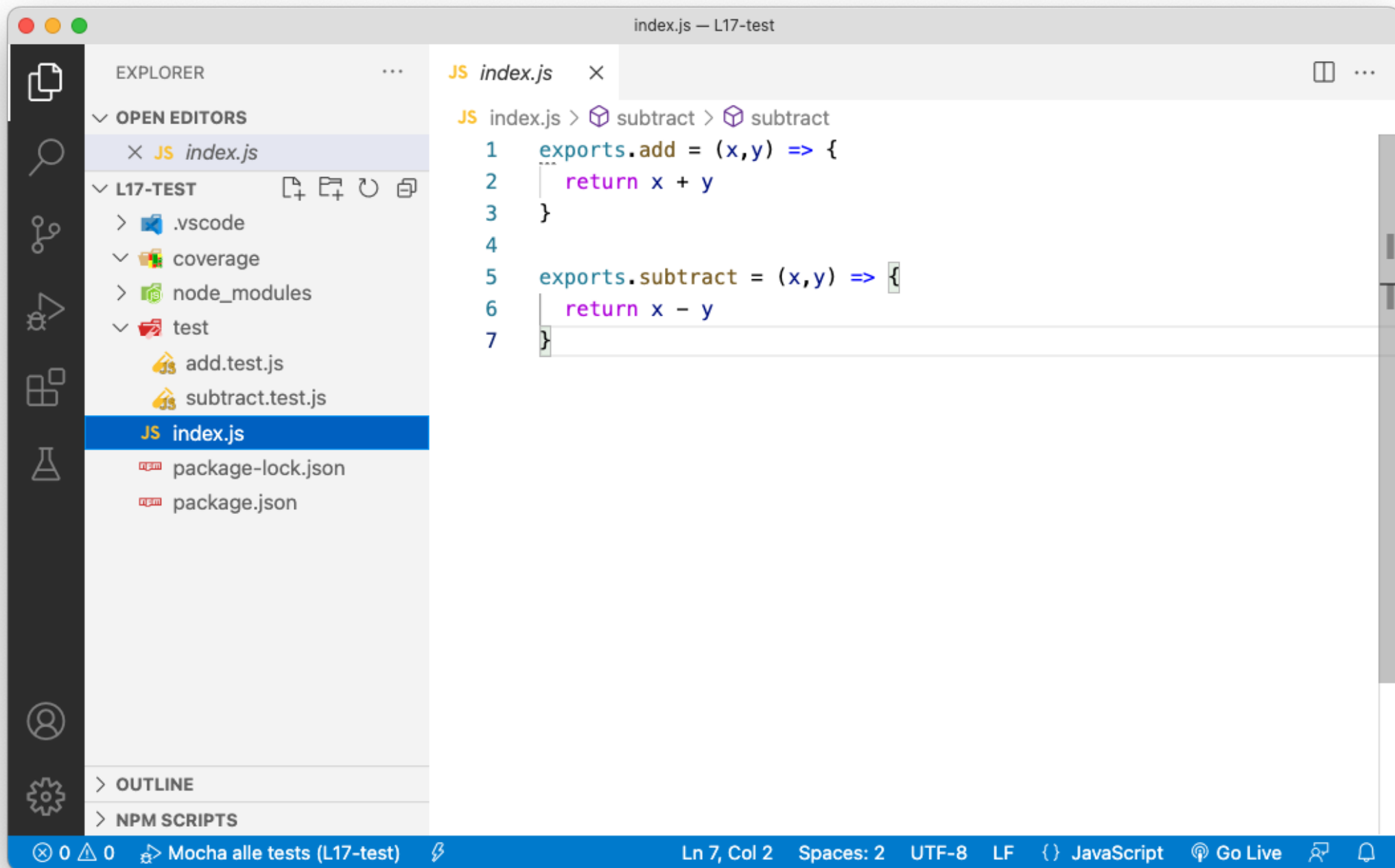
launch.json 1

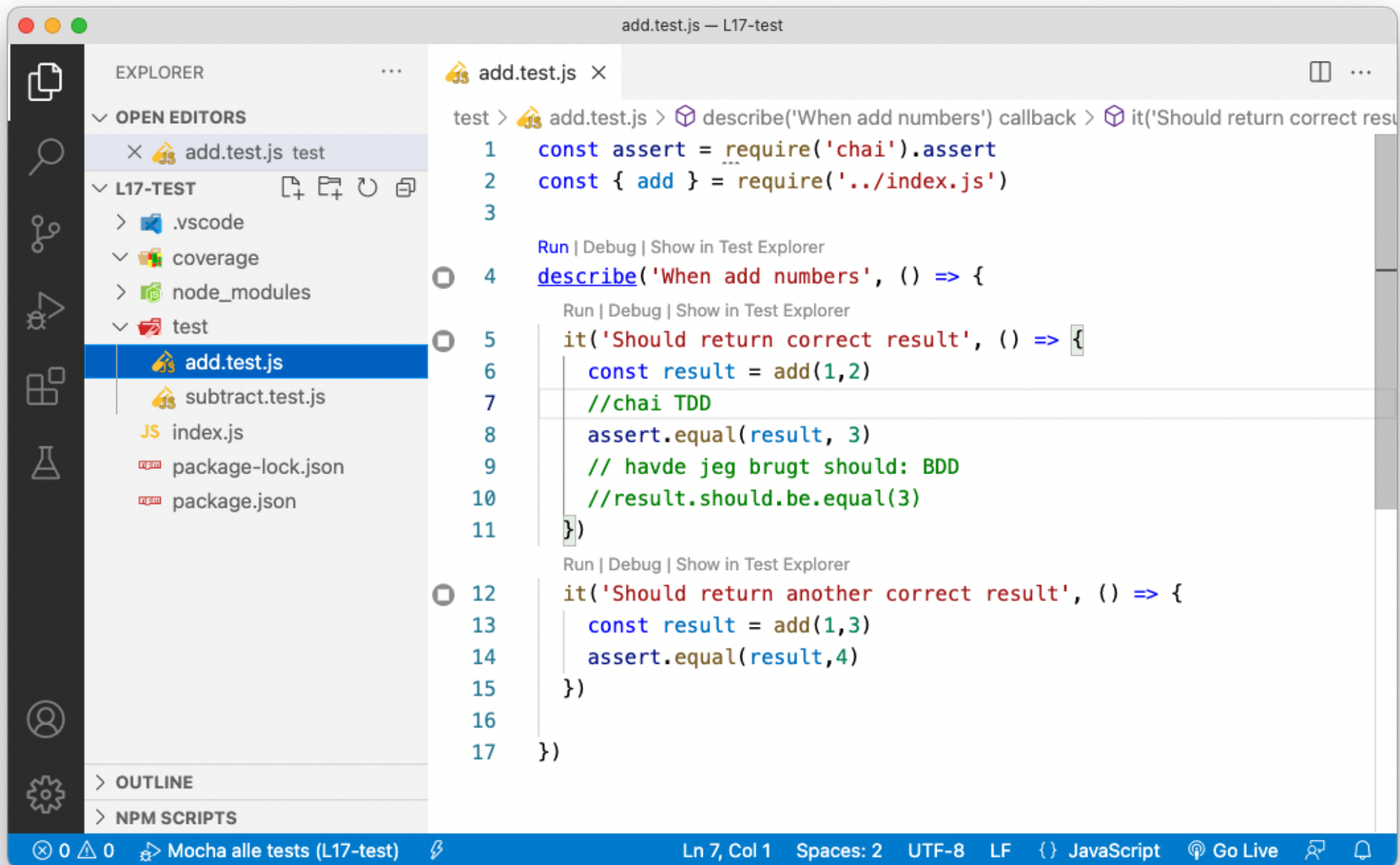
×

.vscode > launch.json > ...

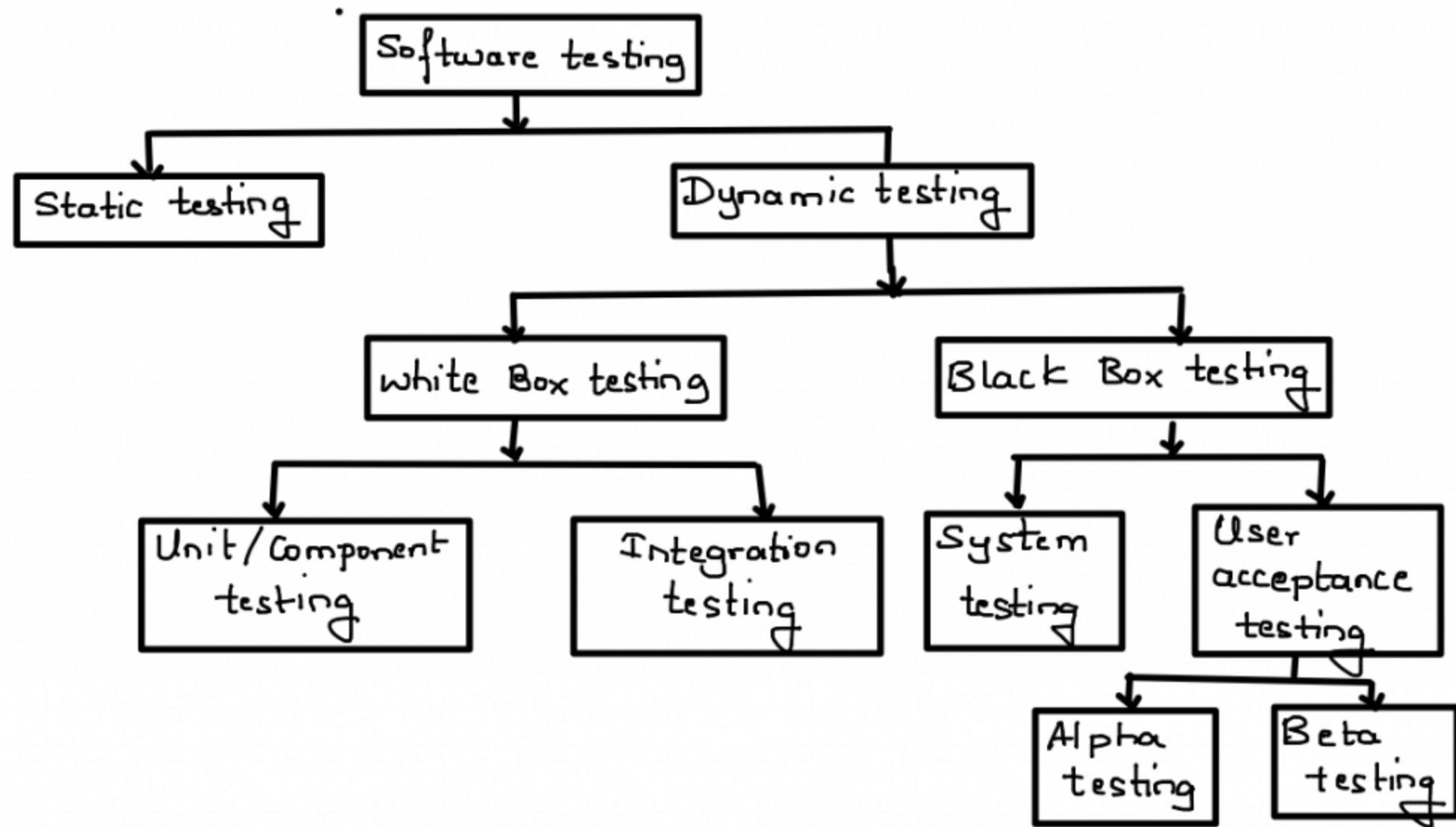
```
1 {
2   // Use IntelliSense to learn about possible attributes.
3   // Hover to view descriptions of existing attributes.
4   // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
5   "version": "0.2.0",
6   "configurations": [
7     {
8       "request": "launch",
9       "name": "Mocha alle tests",
10      "skipFiles": [
11        "<node_internals>/**"
12      ],
13      "program": "${workspaceFolder}/node_modules/mocha/bin/_mocha",
14      "args": [
15        "--timeout",
16        "999999",
17        "${workspaceFolder}/test"
18      ],
19    }
20  ]
21 }
```

Add Configuration...





Hvad er test så?



statisk testing er, hvor vi kikker på fejl i koden i editoren altså syntaks fejl og banale logiske fejl som loops.

Dynamisk testing er runtime fejl.

Unit testing er hvor vi kikker på koden og forfiner den så vi opnår visse kvalitetsstandarder.

Dette leder os til TDD & BDD

Hvad er Unit testing?

En unit er lige hvad vi vil have det skal være 🤪

- en klasse
- en metode
- eller bare en linie kode

Jo mindre jo bedre 😊

Test-driven development TDD betyder at der skrives bestemte test før produktionen for på den måde at få lidt sikkerhed for at koden der produceres virker som den skal.

Behaviour-driven development BDD er en underklasse af test-driven development, hvor vi skriver i simpel og *menneske venlig* stil når testen skrives.

Mocha

Mocha er et light-weight Node.js test framework

Mocha er det, der kører testen, altså describe, it mm.

I JUnit svarer det til at markere en metode eller en klasse som test

Chai

Chai er et TDD assertion bibliotek for Node.js

Chai (eller et andet assertion framework som should) er det, der tester, om det ser godt nok ud efter testen.

I JUnit svarer det f.eks. til `assert.equals`

I gang

i `package.json` skriver du mocha som testsuite:

```
"scripts": {  
  "test": "mocha"  
},
```

i Terminalen skal du installere mocha og chai som dev-dependencies

```
npm i --save-dev mocha chai
```

Eksempel

→ **Lektion 17 – test** npm test

```
> l17test@0.0.1 test  
> mocha
```

Chef test

```
I will like to have: Pav Bhaji  
  ✓ check the dish has valid name.  
I will like to have: Pokoras  
  ✓ check for a dish in menu.
```

2 passing (3ms)