**Тестування**

<https://jestjs.io/docs/en/expect.html>

<http://www.protesting.ru/testing/testtypes.html>

<http://software-testing.ru/>

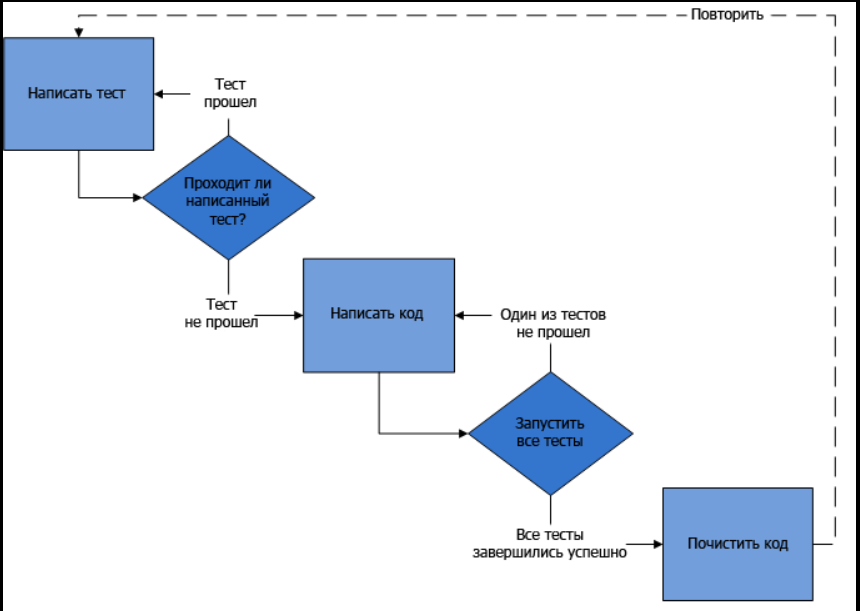
<https://www.osp.ru/os/2008/07/5478839/>

<https://habr.com/ru/post/279535/>

<https://dou.ua/forums/topic/13389/>

**Разработка через [тестирование](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F" \o "Тестирование программного обеспечения)** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA" \o "Английский язык) *test-driven development, TDD*) — техника [разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F" \o "Разработка программного обеспечения), которая основывается на повторении очень коротких циклов разработки: сначала пишется тест, покрывающий желаемое изменение, затем пишется код, который позволит пройти тест, и под конец проводится рефакторинг нового кода к соответствующим стандартам

<https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D1%87%D0%B5%D1%80%D0%B5%D0%B7_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5>



=========================================================================================================================================

**JEST**

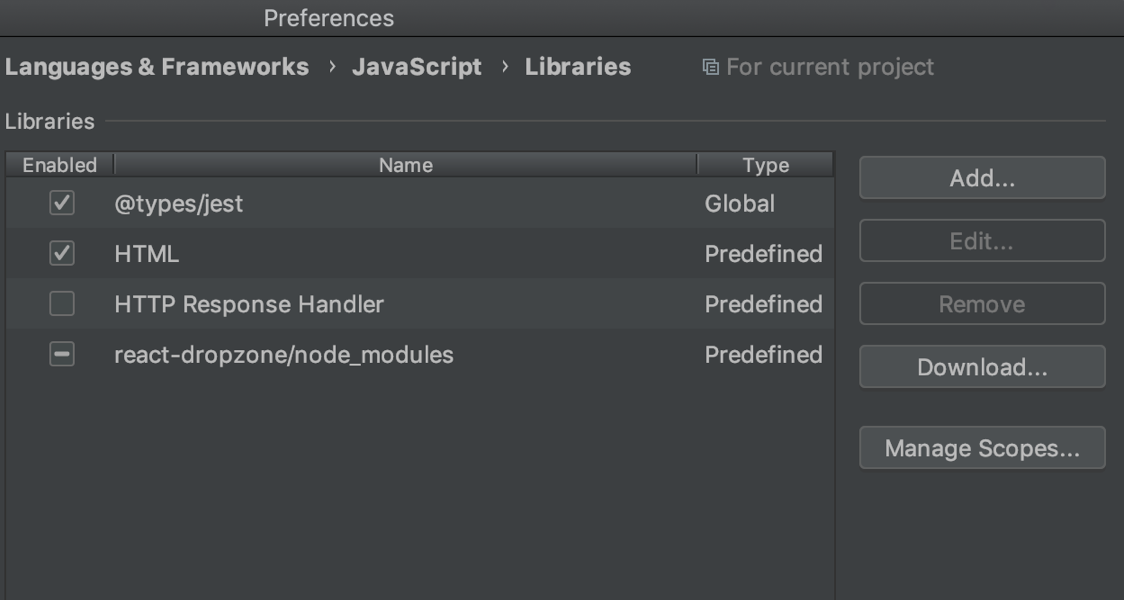
<https://jestjs.io/>

<https://jestjs.io/docs/en/expect.html>

## **Configure code completion**

<https://blog.jetbrains.com/webstorm/2018/10/testing-with-jest-in-webstorm/#configure_code_completion>

You might have noticed that some of the global Jest methods (like describe and beforeEach) in JavaScript files are marked as unresolved in the editor. To fix that, install the TypeScript type definition files for Jest: Go to Preferences | Languages & Frameworks | JavaScript | Libraries, click Download on the right-hand side of the dialog, then search for Jest in the list and click Install. Or add @types/jest to devDependencies in project’s package.json.



This happens because Jest defines these methods in the global scope – so you don’t need to import them in each test file. But it also makes it harder for WebStorm to learn about them from the static analysis of the Jest sources – that’s why they are marked by default as unresolved.

**Тестування функцій**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Маємо файл з функціями | назва файлу, що тестується . js | **------------ MyFile.js--------------**  **function** *sum*(a,b) {  **return** a+b; } **function** *mult*(a,b) {  **return** a\*b; }  ......................... **export** {*sum*,*mult*,*MyFunc*} |
| Створюємо файл з тестами (до назви файлу з кодом додаємо “**.test**”) | назва файлу, що тестується .**test** .js | **----------------- MyFile.test.js--------------**  **import** {*sum*, *mult*, *MyFunc*} **from './MyFile'** describe(**"Tests of my functions "**, () => {  //- - - - - - - - - - - - - - - - - -  it(**"test sum 1"**, () => {  expect(*sum*(2, 3)).toEqual(5)  })  //- - - - - - - - - - - - - - - - - -  it(**"test sum 2"**, () => {  expect(*sum*()).toEqual(**NaN**)  })  //- - - - - - - - - - - - - - - - - -  it(**"test** *mult***"**, () => {  expect(*mult*(5, 3)).toBeGreaterThan(10)  })  //- - - - - - - - - - - - - - - - - -  it(**"Test Mu func"**, () => {  expect(*MyFunc*(5)).toMatchSnapshot()  })  } ); |
| Описуємо групу, що буде містити тести | describe(назва групи текстів , ()=> {  ...  опис тестів  ...  }) |
| Описуємо тести | it(текстовий опис тесту, () => {  **expect**( *вираз для тестування* ).toEqual( очікуване значення )    })  Замість toEqual можна використовувати багато інших методів  <https://jestjs.io/docs/en/expect.html> |
| Запуск тестів | npm test |  |

Методи для порівняння результату з очікуваним значення при тестуванні

<https://jestjs.io/docs/en/expect.html>

* [expect(value)](https://jestjs.io/docs/en/expect.html#expectvalue)
* [expect.extend(matchers)](https://jestjs.io/docs/en/expect.html#expectextendmatchers)
* [expect.anything()](https://jestjs.io/docs/en/expect.html#expectanything)
* [expect.any(constructor)](https://jestjs.io/docs/en/expect.html#expectanyconstructor)
* [expect.arrayContaining(array)](https://jestjs.io/docs/en/expect.html#expectarraycontainingarray)
* [expect.assertions(number)](https://jestjs.io/docs/en/expect.html#expectassertionsnumber)
* [expect.hasAssertions()](https://jestjs.io/docs/en/expect.html#expecthasassertions)
* [expect.not.arrayContaining(array)](https://jestjs.io/docs/en/expect.html#expectnotarraycontainingarray)
* [expect.not.objectContaining(object)](https://jestjs.io/docs/en/expect.html#expectnotobjectcontainingobject)
* [expect.not.stringContaining(string)](https://jestjs.io/docs/en/expect.html#expectnotstringcontainingstring)
* [expect.not.stringMatching(string | regexp)](https://jestjs.io/docs/en/expect.html#expectnotstringmatchingstring-regexp)
* [expect.objectContaining(object)](https://jestjs.io/docs/en/expect.html#expectobjectcontainingobject)
* [expect.stringContaining(string)](https://jestjs.io/docs/en/expect.html#expectstringcontainingstring)
* [expect.stringMatching(string | regexp)](https://jestjs.io/docs/en/expect.html#expectstringmatchingstring-regexp)
* [expect.addSnapshotSerializer(serializer)](https://jestjs.io/docs/en/expect.html#expectaddsnapshotserializerserializer)
* [.not](https://jestjs.io/docs/en/expect.html#not)
* [.resolves](https://jestjs.io/docs/en/expect.html#resolves)
* [.rejects](https://jestjs.io/docs/en/expect.html#rejects)
* [.toBe(value)](https://jestjs.io/docs/en/expect.html#tobevalue)
* [.toHaveBeenCalled()](https://jestjs.io/docs/en/expect.html#tohavebeencalled)
* [.toHaveBeenCalledTimes(number)](https://jestjs.io/docs/en/expect.html#tohavebeencalledtimesnumber)
* [.toHaveBeenCalledWith(arg1, arg2, ...)](https://jestjs.io/docs/en/expect.html#tohavebeencalledwitharg1-arg2-)
* [.toHaveBeenLastCalledWith(arg1, arg2, ...)](https://jestjs.io/docs/en/expect.html#tohavebeenlastcalledwitharg1-arg2-)
* [.toHaveBeenNthCalledWith(nthCall, arg1, arg2, ....)](https://jestjs.io/docs/en/expect.html#tohavebeennthcalledwithnthcall-arg1-arg2-)
* [.toHaveReturned()](https://jestjs.io/docs/en/expect.html#tohavereturned)
* [.toHaveReturnedTimes(number)](https://jestjs.io/docs/en/expect.html#tohavereturnedtimesnumber)
* [.toHaveReturnedWith(value)](https://jestjs.io/docs/en/expect.html#tohavereturnedwithvalue)
* [.toHaveLastReturnedWith(value)](https://jestjs.io/docs/en/expect.html#tohavelastreturnedwithvalue)
* [.toHaveNthReturnedWith(nthCall, value)](https://jestjs.io/docs/en/expect.html#tohaventhreturnedwithnthcall-value)
* [.toBeCloseTo(number, numDigits)](https://jestjs.io/docs/en/expect.html#tobeclosetonumber-numdigits)
* [.toBeDefined()](https://jestjs.io/docs/en/expect.html#tobedefined)
* [.toBeFalsy()](https://jestjs.io/docs/en/expect.html#tobefalsy)
* [.toBeGreaterThan(number)](https://jestjs.io/docs/en/expect.html#tobegreaterthannumber)
* [.toBeGreaterThanOrEqual(number)](https://jestjs.io/docs/en/expect.html#tobegreaterthanorequalnumber)
* [.toBeLessThan(number)](https://jestjs.io/docs/en/expect.html#tobelessthannumber)
* [.toBeLessThanOrEqual(number)](https://jestjs.io/docs/en/expect.html#tobelessthanorequalnumber)
* [.toBeInstanceOf(Class)](https://jestjs.io/docs/en/expect.html#tobeinstanceofclass)
* [.toBeNull()](https://jestjs.io/docs/en/expect.html#tobenull)
* [.toBeTruthy()](https://jestjs.io/docs/en/expect.html#tobetruthy)
* [.toBeUndefined()](https://jestjs.io/docs/en/expect.html#tobeundefined)
* [.toBeNaN()](https://jestjs.io/docs/en/expect.html#tobenan)
* [.toContain(item)](https://jestjs.io/docs/en/expect.html#tocontainitem)
* [.toContainEqual(item)](https://jestjs.io/docs/en/expect.html#tocontainequalitem)
* [.toEqual(value)](https://jestjs.io/docs/en/expect.html#toequalvalue)
* [.toHaveLength(number)](https://jestjs.io/docs/en/expect.html#tohavelengthnumber)
* [.toMatch(regexpOrString)](https://jestjs.io/docs/en/expect.html#tomatchregexporstring)
* [.toMatchObject(object)](https://jestjs.io/docs/en/expect.html#tomatchobjectobject)
* [.toHaveProperty(keyPath, value)](https://jestjs.io/docs/en/expect.html#tohavepropertykeypath-value)
* [.toMatchSnapshot(propertyMatchers, snapshotName)](https://jestjs.io/docs/en/expect.html#tomatchsnapshotpropertymatchers-snapshotname)
* [.toMatchInlineSnapshot(propertyMatchers, inlineSnapshot)](https://jestjs.io/docs/en/expect.html#tomatchinlinesnapshotpropertymatchers-inlinesnapshot)
* [.toStrictEqual(value)](https://jestjs.io/docs/en/expect.html#tostrictequalvalue)
* [.toThrow(error)](https://jestjs.io/docs/en/expect.html#tothrowerror)
* [.toThrowErrorMatchingSnapshot()](https://jestjs.io/docs/en/expect.html#tothrowerrormatchingsnapshot)
* [.toThrowErrorMatchingInlineSnapshot()](https://jestjs.io/docs/en/expect.html#tothrowerrormatchinginlinesnapshot)

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **toBe** | проверяет на строгое соответствие |  |
| **toEqual** | рекурсивно проверяет соответствие, полезно для сравнения массивов и объектов. Сравнение происходит по значениям полей, а не ссылкам самих значений. |  |
| **toBeNull** | проверка на соответствие значения к null |  |
| **toBeUndefined** | проверка на соответствие undefined |  |
| **toBeDefined** | проверка на наличие метода или поля объекта. Проверяется на то, что значение не undefined |  |
| **toBeTruthly** | проверка на соответствие true |  |
| **toBeFalsy** | проверка на соответствие false |  |
| **toBeGreaterThan** | проверка на то, что число больше |  |
| **toBeGreaterThanOrEqual** | проверка на больше или равно |  |
| **toBeLessThan** | проверка на то, что число меньше |  |
| **toBeLessThanOrEqual** | проверка на меньше или равно |  |
| **toBeCloseTo** | сравнение с указанием округления, вторым аргументом.  Все потому что 0.2 + 0.1 = 0.30000000000000004 |  |
| **toMatch** | сравнение строки с помощью регулярного выражения |  |
| **toContain** | проверка на наличие элемента в массиве |  |

**Snapshot**

Якщо вираз для тестування дуже складний і важко визначити результат, який повинен бути при заданих параметрах, то можна використати метод toMatchSnapshot при використанні якого у перший раз при запуску тесту запам’ятовується значення функції. Це значення функції буде використовуватися як правильне значення при наступних запусках. Якщо необхідно оновити значення (якщо ми змінили функцію), то необхідно натиснути клавішу “u”

<https://jestjs.io/docs/en/snapshot-testing>

|  |  |
| --- | --- |
|  | -------------------- функція ---------------------  **function** *MyFunc*(x) {  **return *Math***.floor(10\****Math***.sin(x)+311\****Math***.cos(x)); }  -------------------- тест ------------------------  it(**"Test Mu func"**,()=>{  expect(*MyFunc*(5)).**toMatchSnapshot**() }) |
| it('текстовий опис тесту', () => {  expect( вираз ).toMatchSnapshot();  }); | **import** React **from** 'react';  **import** Link **from** '../Link.react';  **import** renderer **from** 'react-test-renderer';  it('renders correctly', () => {  **const** tree = renderer  .create(<**Link** page="http://www.facebook.com">Facebook</**Link**>)  .toJSON();  expect(tree).toMatchSnapshot();  }); |
|  |  |

Enzyme

<https://github.com/airbnb/enzyme>

<https://airbnb.io/enzyme/docs/api/>

**Встановлення**

npm i --save-dev enzyme enzyme-adapter-react-16

**Конфігурація**

П при використанні Create React App необхідно створити файл src/setupTests.js (він автоматично буде виконуватися при запуску тестів)

-------------- src/setupTests.js-------------

import Enzyme from 'enzyme';

import Adapter from 'enzyme-adapter-react-16';

Enzyme.configure({ adapter: new Adapter() });

***Альтернативні варіанти підключення***

The file 'setupTests' has to be imported to the test file:

import MessageBox from "../MessageBox";

import { shallow } from 'enzyme';

import React from 'react';

import "../setupTests" 🡨 Підключення конфігураційного файлу

----------------------------------------------------------------------------------------------

Also, if you don't want to import your setupTests.js file into every test file, you can place it in your package.json folder:

"jest": {

"setupTestFrameworkScriptFile": "./test/setupTests.js" }

Enzyme — бібліотека для тестування react компонентів. При цьому можуть бути використані такі методи для роботи з компонатами:

shallow (<https://airbnb.io/enzyme/docs/api/shallow.html>) – дозволяє створити поверхневий екземпляр компонента (тобто, якщо використовуються зовнішні компоненти, то вони рендеритися не будуть)

mount (<https://airbnb.io/enzyme/docs/api/mount.html>) – дозволяє створити повноцінний екземпляр компонента (з генерацією також зовнішніх компонентів, які використовуються у компоненті, що тестується)

render(<https://airbnb.io/enzyme/docs/api/render.html>) – дозволяє одержати згенеровану HTML розмітку (використовується бібліотека  [Cheerio](http://cheeriojs.github.io/cheerio/" \t "_blank))

======================================================================

Приклад згенерованих екземплярів компонента з використанням різних методів (для виведення згенерованого екземпляра можна скористатися методом debug).

|  |  |  |
| --- | --- | --- |
| Компонент |  |  |
| **import** React **from 'react'**; //------------------------------------ **const** *First* = () => {  **return** (  <**div**>  <**h1**>Перша частина (show==true)</**h1**>  <**button**> Click me</**button**>  </**div**>  ) }  //------------------------------------ **const** *Second* = () => {  **return** (  <**div**>  <**h1**>Друга частина (show==false)</**h1**>  <**p**>Дуже короткий опис</**p**>  <**button**> Click me</**button**>  </**div**>  ) }  //==================================== **const** *OuterComp* = ({show}) => {  **return** (  <**div**>  {show ?  <**First**/>  :  <**Second**/>  }  </**div**>  ); };  **export default** *OuterComp*; | **------------ shallow ----------------**  **import** {**shallow**} **from 'enzyme'**;  **import** React **from 'react'**;  **import** *OuterComp* **from './testComp' import '../setupTests'** describe(**"Тестк компонента"**,()=>{  it(**"Тести випадку show=true"**,()=>{ **const** wrapper = **shallow**( <**OuterComp show**/>)  ***console***.log(wrapper.*debug*())  *//expect(wrapper.contains(p)).to* })   }) |  |
| **----------- mount -----------------**  **import** { **mount**} **from 'enzyme'**;  **import** React **from 'react'**;  **import** *OuterComp* **from './testComp' import '../setupTests'** describe(**"Тестк компонента"**,()=>{  it(**"Тести випадку show=true"**,()=>{ **const** wrapper = **mount**( <**OuterComp show**/>)  ***console***.log(wrapper.*debug*())  *//expect(wrapper.contains(p)).to* }) }) |  |
| **----------------- render ------------**  **import** { **render**} **from 'enzyme'**;  **import** React **from 'react'**; **import** *OuterComp* **from './testComp'** describe(**"Тестк компонента"**,()=>{  it(**"Тести випадку show=true"**,()=>{  **const** wrapper = **render**( <**OuterComp show**/>)  ***console***.log(wrapper.html())  *//expect(wrapper.contains(p)).to* }) }) | Що одержали:  <div><h1>&#x41F;&#x435;&#x440;&#x448;&#x430; &#x447;&#x430;&#x441;&#x442;&#x438;&#x43D;&#x430; (show==true)</h1><button> Click me</button></div>  Відформатований аналог: |

**shallow**

(<https://airbnb.io/enzyme/docs/api/shallow.html>)

## shallow(node[, options]) => ShallowWrapper

#### Arguments

1. node (ReactElement): The node to render
2. options (Object [optional]):
   * options.context: (Object [optional]): Context to be passed into the component
   * options.disableLifecycleMethods: (Boolean [optional]): If set to true, componentDidMount is not called on the component, and componentDidUpdate is not called after [setProps](https://airbnb.io/enzyme/docs/api/ShallowWrapper/setProps.html) and [setContext](https://airbnb.io/enzyme/docs/api/ShallowWrapper/setContext.html). Default to false.

#### Returns

ShallowWrapper: The wrapper instance around the rendered output.

|  |  |
| --- | --- |
| Загальна форма | Приклад |
| import { shallow } from 'enzyme';  it('текстовий опис тесту', () => {  // Створюємо поверхневий екземпляр ()  const wrapper = shallow(компонент [, options]);  // Перевіряємо поверхневий екземпляр на відповідність  expect(wrapper.метод пошуку/фільтрації('селектор')).правило перевірки;  }); | **import** {**shallow**, **mount**, **render**} **from 'enzyme'**;  **import** React **from 'react'**; **import** *OuterComp*, {*First*,*Second*} **from '../testComp'** describe(**"renders without crashingя"**,()=>{  it(**"Тести випадку show=true"**,()=>{  **const** wrapper = **shallow**( <**OuterComp show**/>)   })  it(**"Тести випадку show=false"**,()=>{  **const** wrapper = **shallow**( <**OuterComp**/>)  }) })  describe(**"Нявність дочірніх компонетів"**,()=>{  it(**"Наявність компонента <First/>"**,()=>{  **const** wrapper = **shallow**( <**OuterComp show**/>)  expect(wrapper.contains(<**First**/>)).toBeTruthy();  })  it(**"Наявність компонента <Second/>"**,()=>{  **const** wrapper = **shallow**( <**OuterComp** />)  expect(wrapper.contains(<**Second text="My second component"**/>)).toBeTruthy();  }) })  describe(**"Нявність атрибутів"**,()=>{  it(**"Наявність компонента Second"**,()=>{  **const** wrapper = **shallow**( <**OuterComp** />)  expect(wrapper.find(*Second*)).toHaveLength(1);  })  it(**"Наявність** prop(**'text'**)**"**,()=>{  **const** wrapper = **shallow**( <**OuterComp** />)  expect(wrapper.find(*Second*).prop(**'text'**)).toBeDefined();  })  it(**"Перевірка значення властивості** prop(**'text'**)**"**,()=>{  **const** wrapper = **shallow**( <**OuterComp** />)  expect(wrapper.find(*Second*).prop(**'text'**)).toMatchSnapshot();  })  }) |

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**mount**

<https://airbnb.io/enzyme/docs/api/mount.html>

## mount(node[, options]) => ReactWrapper

#### Arguments

1. node (ReactElement): The node to render
2. options (Object [optional]):
3. options.context: (Object [optional]): Context to be passed into the component
4. options.attachTo: (DOMElement [optional]): DOM Element to attach the component to.
5. options.childContextTypes: (Object [optional]): Merged contextTypes for all children of the wrapper.

#### Returns

ReactWrapper: The wrapper instance around the rendered output.

|  |  |
| --- | --- |
| Загальна форма | Приклад |
| import { **mount** } from 'enzyme';  it('текстовий опис тесту', () => {  // Створюємо поверхневий екземпляр ()  const wrapper = **mount** (компонент [, options]);  // Перевіряємо поверхневий екземпляр на відповідність  expect(wrapper.метод пошуку/фільтрації('селектор')).правило перевірки(1);  }); | **import** {**shallow**, **mount**, **render**} **from 'enzyme'**;  **import** React **from 'react'**; **import** *OuterComp*, {*First*, *Second*} **from '../testComp'  *describe***(**"Нявність компонента"**, () => {  ***it***(**"Наявність компонента <Second/>"**, () => {  **const** wrapper = **mount**(<**OuterComp**/>)  ***expect***(wrapper.find(*Second*)).toHaveLength(1);  })  ***it***(**"Наявність компонента h1"**, () => {  **const** wrapper = **mount**(<**OuterComp**/>)  ***expect***(wrapper.find(**'h1'**)).toBeDefined();  })  ***it***(**"Наявність компонента p з класом my-p"**, () => {  **const** wrapper = **mount**(<**OuterComp**/>)  ***expect***(wrapper.find(**'.my-p'**)).toBeDefined();  })  ***it***(**"Наявність компонента з id='start'"**, () => {  **const** wrapper = **mount**(<**OuterComp**/>)  ***expect***(wrapper.find(**'#start'**)).toBeDefined();  })  ***it***(**"Наявність компонента з атрибутом [author=\"Georg\"]"**, () => {  **const** wrapper = **mount**(<**OuterComp**/>)  ***expect***(wrapper.find(**'[author="Georg"]'**)).toBeDefined();  })   ***it***(**"Підрахунок кількості дочірніх"**, () => {  **const** wrapper = **mount**(<**OuterComp**/>)  ***expect***(wrapper.find(**'div'**).children().**length**).toEqual(6)  }) }) |

render

<https://airbnb.io/enzyme/docs/api/render.html>

|  |  |
| --- | --- |
|  |  |
| import { render } from 'enzyme';  it('текстовий опис тесту', () => {  // Створюємо поверхневий екземпляр ()  const wrapper = **render** (компонент [, options]);  // Перевіряємо поверхневий екземпляр на відповідність  expect(wrapper.метод пошуку/фільтрації('селектор')).правило перевірки(1);  }); | mport React from 'react';  import { render } from 'enzyme';  import PropTypes from 'prop-types';  describe('<Foo />', () => {  it('renders three `.foo-bar`s', () => {  const wrapper = render(<Foo />);  expect(wrapper.find('.foo-bar')).to.have.lengthOf(3);  });  it('rendered the title', () => {  const wrapper = render(<Foo title="unique" />);  expect(wrapper.text()).to.contain('unique');  });  it('can pass in context', () => {  function SimpleComponent(props, context) {  const { name } = context;  return <div>{name}</div>;  }  SimpleComponent.contextTypes = {  name: PropTypes.string,  };  const context = { name: 'foo' };  const wrapper = render(<SimpleComponent />, { context });  expect(wrapper.text()).to.equal('foo');  });  }); |

**Cелектори**

<https://airbnb.io/enzyme/docs/api/selector.html>

**enzyme Selectors**

Many methods in enzyme’s API accept a *selector* as an argument. Selectors in enzyme can fall into one of the following five categories:

**1. A Valid CSS Selector**

enzyme supports a subset of valid CSS selectors to find nodes inside a render tree. Support is as follows:

* class syntax (.foo, .foo-bar, etc.)
* element syntax (input, div, span, etc.)
* id syntax (#foo, #foo-bar, etc.)
* attribute syntax ([href="foo"], [type="text"], and the other attribute selectors listed [here](https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/Attribute_selectors).)

Further, enzyme supports combining any of those supported syntaxes together to uniquely identify a single node. For instance:

div.foo.bar

input#input-name

a[href="foo"]

enzyme also gives support for the following contextual selectors

.foo .bar

.foo > .bar

.foo + .bar

.foo ~ .bar

.foo input

**Want more CSS support?**

PRs implementing more support for CSS selectors will be accepted and is an area of development for enzyme that will likely be focused on in the future.

**2. Prop Selector**

In addition to traditional CSS selectors, enzyme supports using a React prop like an Attribute Selector as if it were an HTML attribute. Strings, Numbers, and Boolean property values are supported.

const wrapper = mount((

<div>

<span foo={3} bar={false} title="baz" />

</div>

));

wrapper.find('[foo=3]');

wrapper.find('[bar=false]');

wrapper.find('[title="baz"]');

**The Key and Ref Prop**

While in most cases, any React prop can be used, there are exceptions. The key and ref props will never work. This decision comes from how React uses these props internally, which means they should not be relied upon.

**3. A React Component Constructor**

enzyme allows you to find components based on their constructor. You can pass in the reference to the component’s constructor:

function MyComponent() {

return <div />;

}

// find instances of MyComponent

const myComponents = wrapper.find(MyComponent);

**4. A React Component’s displayName**

enzyme allows you to find components based on a component’s displayName. If a component exists in a render tree where its displayName is set and has its first character as a capital letter, a string can be used to find it:

function MyComponent() {

return <div />;

}

MyComponent.displayName = 'My Component';

// find instances of MyComponent

const myComponents = wrapper.find('My Component');

NOTE: This will *only* work if the selector (and thus the component’s displayName) is a string starting with a capital letter. Strings starting with lower case letters will assume it is a CSS Selecting a HOC-wrapped component, or a component with a custom displayName, even with lowercase letters (for example, withHOC(MyComponent)) will work as well. selector using the tag syntax.

**5. Object Property Selector**

enzyme allows you to find components and nodes based on a subset of their properties:

const wrapper = mount((

<div>

<span foo={3} bar={false} title="baz" />

</div>

));

wrapper.find({ foo: 3 });

wrapper.find({ bar: false });

wrapper.find({ title: 'baz' });

**Undefined Properties**

Undefined properties are not allowed in the object property selector and will cause an error:

wrapper.find({ foo: 3, bar: undefined });

// => TypeError: Enzyme::Props can't have 'undefined' values. Try using 'findWhere()' instead.

If you have to search by undefined property value, use [.findWhere()](https://airbnb.io/enzyme/docs/api/ShallowWrapper/findWhere.html).