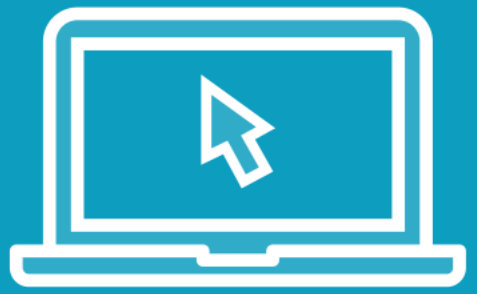
Testing Basic Functionality with Jest

Daniel Stern

Github – danielstern | @danieljackstern

“Testing leads to failure, and failure leads to understanding.”

Demo

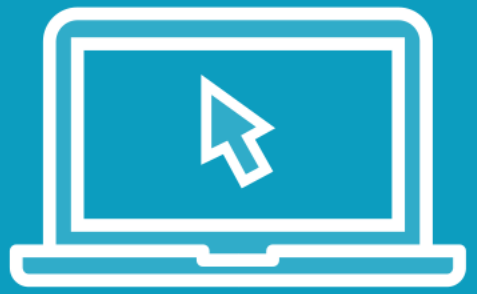


Create `__tests__` folder

Create `__mocks__` folder

Entries must be added to `package.json`

Demo

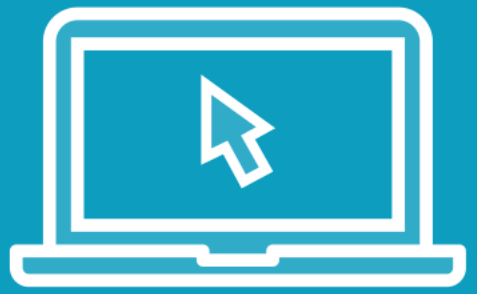


JavaScript files in `__tests__` run as tests

Each module should have a corresponding file with tests

Create a test file for one of the app's modules and scaffolding for tests

Demo



Files are automatically mocked

Modules are automatically mocked

Verify our module is being mocked and note the consequences for our code

Demo

Generally it is desirable not to mock the module you are testing

Implementation details should also not be mocked

Disable mocking of target module in a test file

Test the module

Demo

jQuery and React will not be mocked

External libraries are often covered in their own test suites

A cautiously optimistic expectation that external library will function normally is acceptable

`unmockedModulePathPatterns` config property can handle this

We will update our `package.json` according to the above

Demo

`TestUtils` provide a means of querying React's virtual DOM

Find elements by tag, class, etc.

Pass values to a component, render it to the virtual DOM, and test that it has rendered correctly

Testing Asynchronous Functionality

Asynchronous code is extremely common in real-world web applications

Generally using real async requests would be excessively slow / inconsistent with thousands of tests

Actual HTTP requests are time-consuming compared to synchronous test

Automatic mocks are very useful for this, as mocking a module removes any implicit asynchronocity

Demo

Replace an async call with something that returns a controlled value after a specific interval of 1ms