Skills Network

Testing React Components

IBM Developer

IBM

Testing React Components

IBM Developer

Skills Network

What you will learn

Explain what is testing

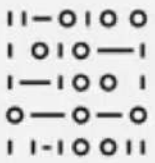
List the advantages and disadvantages of testing

Describe the different approaches of component testing

Describe the testing tools

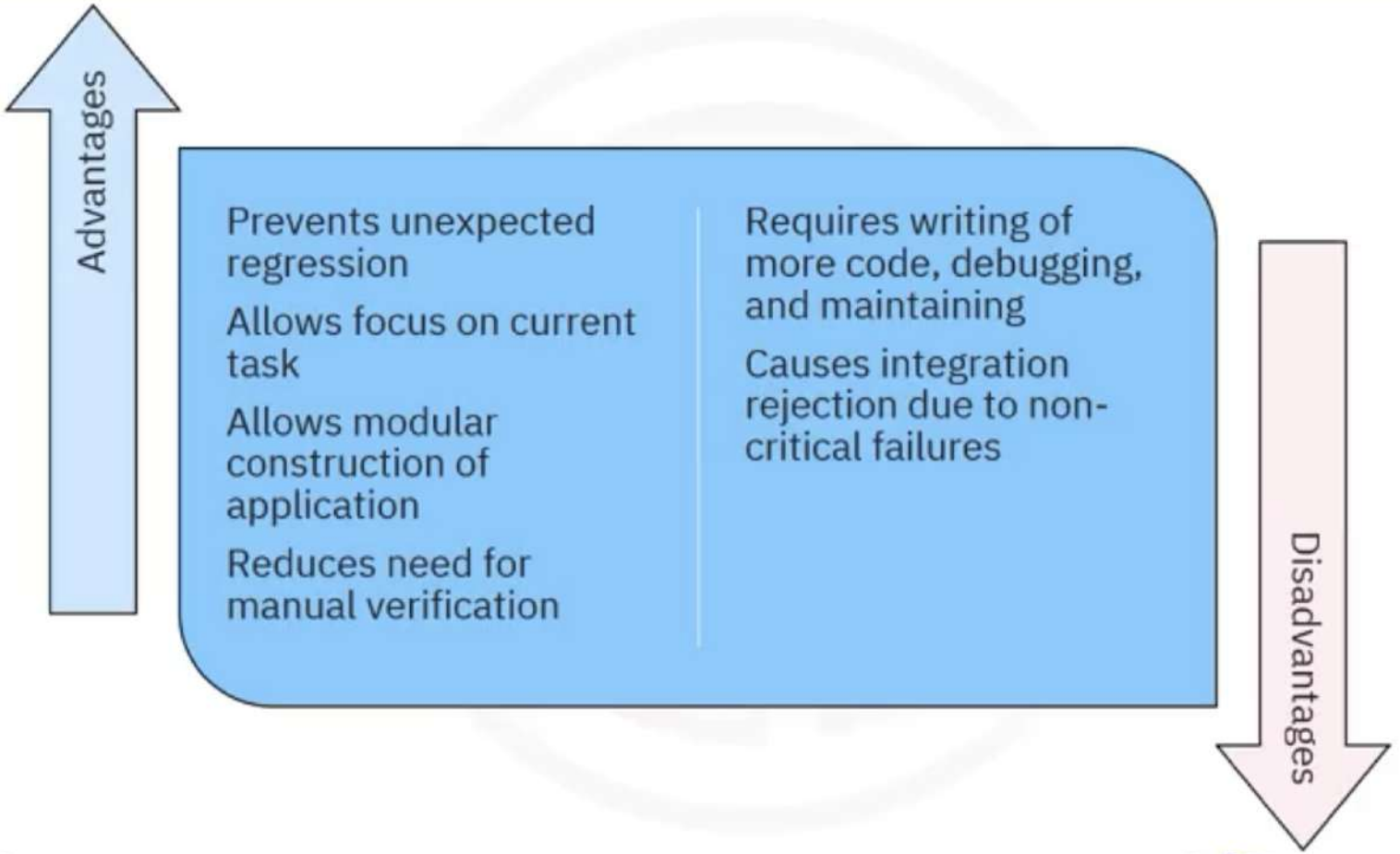
What is testing?

Is a line-by- line review of how your code will execute



Can be a suite of tests comprising bits of code to verify error-free execution of application

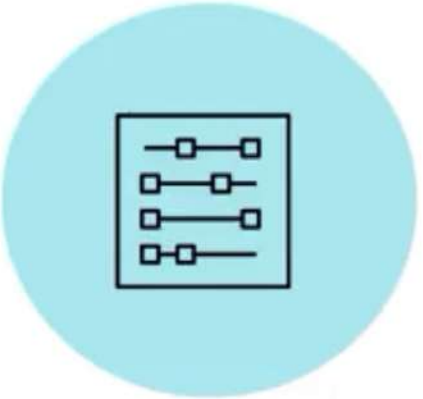
Advantages and disadvantages of testing



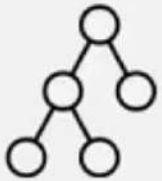
Why test React components?

Testing of React components:

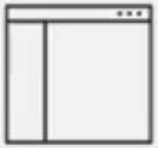
- Ensures application will work as intended
- Verifies the code runs error-free
- Tests functionality by replicating the actions of the end users
- Validates any updates done do not affect the working of the overall application
- Prevents regression that is reappearance of a previous fixed bug



Approaches of React component testing

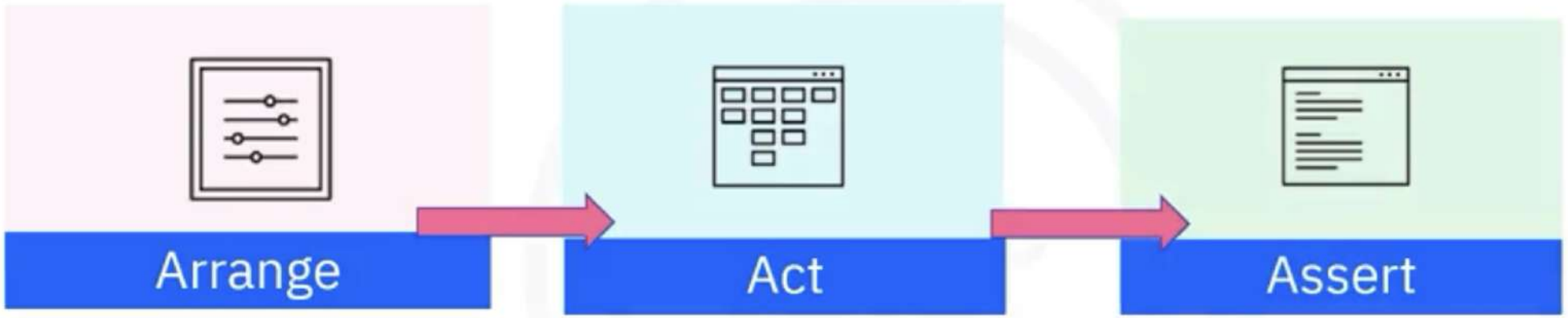


Render component trees in a simple test environment and assert their output



Run application in a realistic browser environment to do end-to-end testing

Phases of React component testing



Choosing the right tool

Speed vs environment

Tools can:

- Give quick feedback but inaccurate browser behavior
- Use real browser environment but give slow iteration

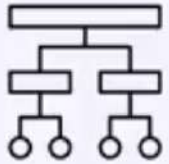
What to mock

- Unit (buttons)
- Integration (form)

React testing libraries/tools

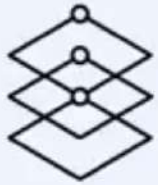
- Mocha: Test runner
- Chai: Assertion library
- Sinon: Spies, stubs, mocks
- Enzyme: Render components

Popular React testing libraries



Jest

Combined Power of Mocha, Chai, Sinon

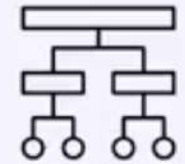


React Testing Library

Set of helpers

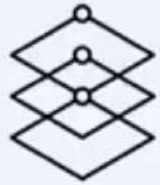
Testing with Jest

- JavaScript test runner
- Combined with Enzyme or React Testing Library
- Can access DOM via jsdom
- Great iteration speed with mocking modules and timers
- Command line tool
- Mock functions
- Offers snapshot
- A dependency to React when using npx-create-react-app



Testing with React Testing Library

- Light utility functions on top of react-dom and react-dom/test-utils
- Facilitate querying the DOM, finding form elements, and finding links and buttons
- Finds elements where the text is not understandable
- Encourages applications to be more accessible
- Is a replacement for Enzyme

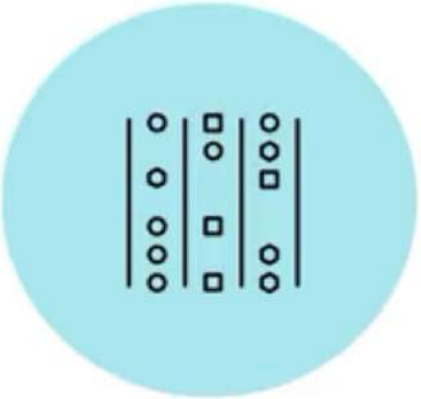


Example of usage of React Testing Library

```
import {render, screen} from '@testing-library/react'
import userEvent from '@testing-library/user-event'
import '@testing-library/jest-dom'
import Fetch from './fetch'

test('loads and displays greeting', async () => {
  // ARRANGE
  render(<Fetch url="/greeting" />)

  // ACT
  await userEvent.click(screen.getByText('Load Greeting'))
  await screen.findByRole('heading')

  // ASSERT
  expect(screen.getByRole('heading')).toHaveTextContent('hello there')
  expect(screen.getByRole('button')).toBeDisabled()
})
```

Benefits of React Test Library

- Provides reliable testing of the components in the same way as an end user would use them
- Enables you to write your tests based on the component output visible to the end users
- Rewriting your tests increases your productivity



Recap

In this video, you learned that:

- Testing is a line-by-line review of how your code will execute.
- You can render component trees in a simple test environment or a realistic browser environment.
- Component tests flow through three phases: Arrange, Act, and Assert.
- React components can be tested using Mocha, Chai, Sinon, Jest and React Testing Library.
- React Testing Library increases confidence in your tests by testing your components in the same way as an end user would use the components.