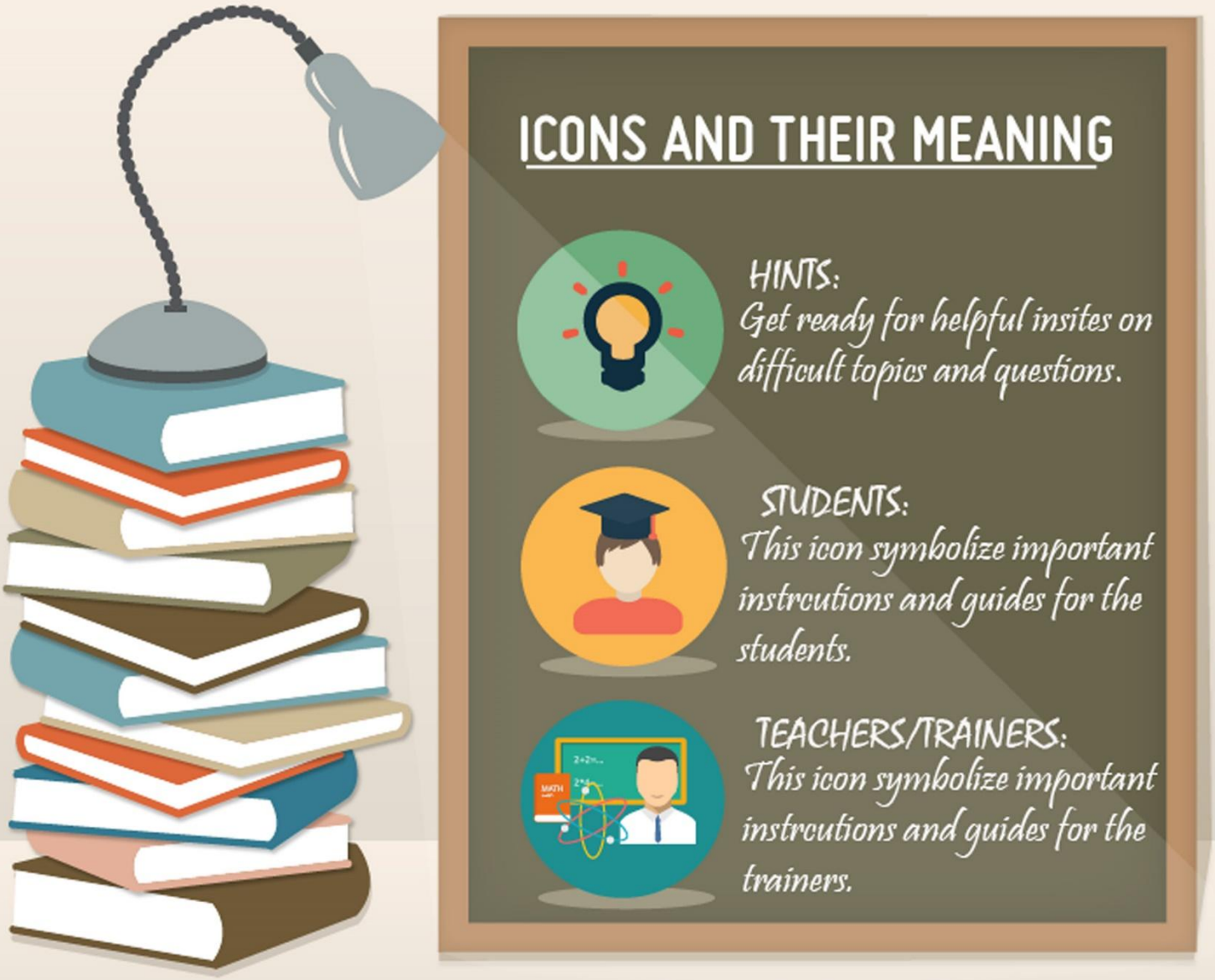
Full Stack Module-IX

Manual V8.3

ANUDIP FOUNDATION


ICONS AND THEIR MEANING
HINTS:
Get ready for helpful insites on difficult topics and questions.
STUDENTS:
This icon symbolize important instrcutions and guides for the students.
TEACHERS/TRAINERS:
This icon symbolize important instrcutions and guides for the trainers.

Total Duration: __ Hours

Lesson 01: Testing React apps with JEST (120 minutes)

Objective: After completing this lesson you will be able to learn about : • JEST introduction • Features • Sample application	**Materials Required:** • Computer With Windows XP and above • Stable Internet connection
Self- Learning Duration: 120 minutes	**Practical Duration:** nil
Total Duration: 120 minutes	

Introduction to Jest Testing Framework

Jest is a JavaScript test runner maintained by Facebook. A test runner is software that looks for tests in your codebase, runs them and displays the results (usually through a CLI interface). Do keep in mind, Jest is a Node-based runner which means that it runs tests in a Node environment as opposed to a real browser. Tests are run within a fake DOM implementation (via jsdom) on the command line.

Features of JEST

- *Performance* - Jest run tests in parallel processes thus minimizing test runtime.
- *Mocking* - Jest allows you to mock objects in your test files. It supports function mocking, manual mocking and timer mocking. You can mock specific objects or turn on automatic mocking with automock which will mock every component/object that the component/object test depends on.
- *Snapshot testing* - When using Jest to test a React or React Native application, you can write a snapshot test that will save the output of a rendered component to file and compare the component's output to the snapshot on subsequent runs. This is useful in knowing when your component changes its behavior.
- *Code coverage support* - This is provided with Jest with no additional packages or configuration.
- *Test isolation and sandboxing* - With Jest, no two tests will ever conflict with each other, nor will there ever be a global or module local state that is going to cause trouble. Sandboxed test files and automatic global state resets for every test.
- *Integrates with other testing libraries* - Jest works well with other testing libraries (e.g. Enzyme, Chai).

Project set up

Before looking at how tests are written, let's first look at the application we'll be testing.

The sample application is a simple countdown timer created in React. To run it, first navigate to the root of the starter project:

$ cd path/to/starter/CountdownTimer

Now, install the necessary libraries:

$ npm install

Run Webpack:

$ webpack

Then run the application with:

$ npm start

Now, simply navigate to http://localhost:3000/ in your browser to observe the outcome.

You can set a time in seconds and start the countdown by clicking on the Start Countdown button.

The functionality of the countdown timer has been separated into three components stored in the app/components folder namely Clock.jsx, Countdown.jsx and CountdownForm.jsx.

The Clock component is responsible for rendering the clock face and formatting the user's input to an MM:SS format.

The CountdownForm component contains a form that takes the user input and passes it to the Countdown component which starts decrementing the value every second, passing the current value to the Clock component for display.

Having looked at the sample application, we'll now proceed with writing tests for it.

First of all, it is about installing and configuring Jest.

Run the following command to install Jest and the babel-jest library which is a Jest plugin for Babel. The application uses Babel for transpiling JSX and ES6 so the plugin is needed for the tests to work.

$ npm install --save-dev jest babel-jest

With babel-jest added, Jest will be able to work with the Babel config file .babelrc to know which presets to run the code through:

```
{
  "presets": ["es2015", "react"]
}
```

The react preset is used to transform JSX into JavaScript and es2015 is used to transform ES6 JavaScript to ES5. Once completed, let's proceed with the first test:

Jest looks for tests to run using the following conventions:

- Files with .test.js suffix.
- Files with .spec.js suffix.
- Files with .js suffix inside a folder named tests.

Other than .js files, it also automatically considers files and tests with the jsx extension.

For our project, we'll store the test files inside a tests folder. In the app folder, create a folder named __tests__.

For the first test, we'll write a simple test that ensures that Jest was set up correctly and that it can run a test successfully.

Create a file inside the app/__tests__ folder, name it app.test.jsx and add the following to the file:

```
describe('App', () => {
it('should be able to run tests', () => {
expect(1 + 2).toEqual(3);
});
});
```

To create a test, you place its code inside an it() or test() block, including a label for the test. You can optionally wrap your tests inside describe() blocks for logical grouping.

Jest comes with a built-in expect() global function for making assertions. The above test checks if the expression 1 + 2 is equal to 3.

Next, modify the test property of the package.json file as shown.

"test": "jest"

You can now run the added test with npm test and see the results in the Terminal.

Reviewing the chapter

- Jest is a JavaScript test runner maintained by Facebook.
- A test runner is software that looks for tests in your codebase, runs them and displays the results.

Testing your skills

1. How many types of MOCKING is supported by JEST

a) 2, b) 3, c) 4, d) 5

2. Which of the following are types of MOCKING in JEST

a) Timer, b) Function, c) Manual, d) All of the above

3. Which of the following are testing libraries

a) JEST, b) Enzyme, c) Chai, d) All of the above

4. _____ is provided with Jest with no additional packages or configuration

a) Code Coverage Support, b) Snapshot testing, c) Mocking, d) Sandboxing

5. Which of these are features of JEST

a) Code coverage support, b) Snapshot testing, c) Both (a) and (b), d) None of the above