

Course > [Week 3: Client-Side Frameworks for Developing Modular Web Page Components](#) > [Advanced Component Design for React](#) >

React application development - Video 3.7

◀ Previous

✔

✔

Next ▶

React application development - Video 3.7

[Bookmark this page](#)

Video 3.7

PENPWJSX2017-V002900

Testing React Apps

- **Mocha** – widely used test runner (testing framework) used to run JavaScript tests
- **Chai** – assertion library for Behavior Driven Testing
- **Enzyme** – testing utility for React for manipulating and inspecting React Component state and output

Property of Penn Engineering, Chris Murphy

0:00 / 20:23

▶ Speed 1.0x

Start of transcript. Skip to the end.

⌵

⌵

We've been looking at React, which allows us to create modular JavaScript components for use in our web applications. So far, we've been writing these components in our HTML pages. But as the components get bigger, and as we get more of them, the amount of code will start to grow. So how can we separate these components into different files and then include them into our application? Here I'll introduce a tool called Node.

⌵

Video

[Download video file](#)

Transcripts

[Download SubRip \(.srt\) file](#)

[Download Text \(.txt\) file](#)

Notes:

- The App.js file used in the "dogs list" example is available [here](#). You will also need to have [AddDog.js](#), [Dogs.js](#), and [DogItem.js](#) in the "components/" subdirectory of the directory containing App.js.
- At 18:06, we should not only check that `dogs[2].name` was correctly set to "Lola", but we should also check that the image URL and breed attributes were also set correctly. We may also want to check that none of the other dog objects was modified.
- Likewise, at 19:02 we should not only check that the number of dog objects has been reduced to 1, but we should also check that the correct object was removed.

◀ Previous

Next ▶