A book with sample markup

Ken Rimple

Chariot Solutions

Table of Contents

Samples

Taken from our React training course lab manual as an illustration of the features of the AsciiDoc format. For information on our courses visit https://chariotsolutions.com/training.

Text and Code Snippets

1. Using your favorite JavaScript-supported IDE or editor (suggestions: Visual Studio Code, Sublime Text, IntelliJ IDEA/Webstorm, VIM or EMACS), open up the `words` demonstration project in the React projects labs folder provided to you by the instructor. You'll be using this project for the rest of the course.
2. Make sure your IDE supports a command-line interface to your operating system, or set up a command-line shell. Multiple panes are recommended as we launch different commands like servers and testing engines simultaneously.
3. Make sure the "Words" project dependencies are installed by executing:

   ```
   yarn install
   ```

   In the beginning, we've pre-configured the project to mount the `Words` component in the file `Words.js`, located at `src/11-ajax-solution`.

   We did this so that you can see how the application operates once it is built. This should have been done during earlier an course lecture.

4. Now, change the import for the root `<Words>` component in `src/index.js` so that you can work on the first in a series of labs against this game.

   `src/index.js`: *the bootstrap file for our labs*

   ```
   import 'bootstrap/dist/css/bootstrap.css';              ①
   import React from 'react';
   import ReactDOM from 'react-dom';

   import Words from './04-components-and-views-start/Words';    ②

   // import Words from './11-ajax-solution/Words';          ③

   ReactDOM.render(<Words/>, document.getElementById('root'));   ④
   ```

   ① Our application will be built based on the Twitter Bootstrap CSS library

   ② All projects in this set start with a single React component, `<Words>`. In each lab, you uncomment the proper import and comment out the one from the last project.

   ③ Remember to comment out the solution for the final lab before starting your work.

   ④ Take the HTML `div` with the ID of `root` and replace its contents with the `<Words>` component

All of the files needed to work with the lab exist in `src/lab-04-components-and-views-start` and it is enabled by pointing to its root component with an import in this file.

Our build tool, Webpack, can support loading CSS files as part of the target bundle. These CSS files are in-lined as JavaScript elements and distributed in the final target output. The `create-react-app` project starter automatically adds this loader.

You may review solutions in `04-components-and-views-solution` instead if you are unable to complete the assignments. We've done this for every chapter. To run it, simply comment out another `Words` component import and uncomment the solution.

Table

Table 1. This is the table caption

Action	Affected State	State Host	Trigger Host
Cell	Cell in column 2, row 1	Cell	Cell in column 2, row 2

Subscribing to the Observable for the Http call

```
constructor(service: HttpDemoService) {
  service.getData()
         .subscribe(
           (results) => { this.data = results },
           (error) => { alert(`Error loading, ${error}`); },
           () => { console.log('call complete'));
}
```

DIAGRAMS

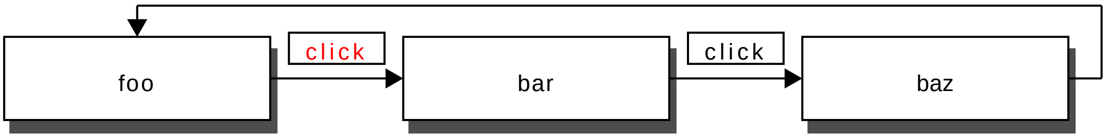

Figure 1. blockdiag

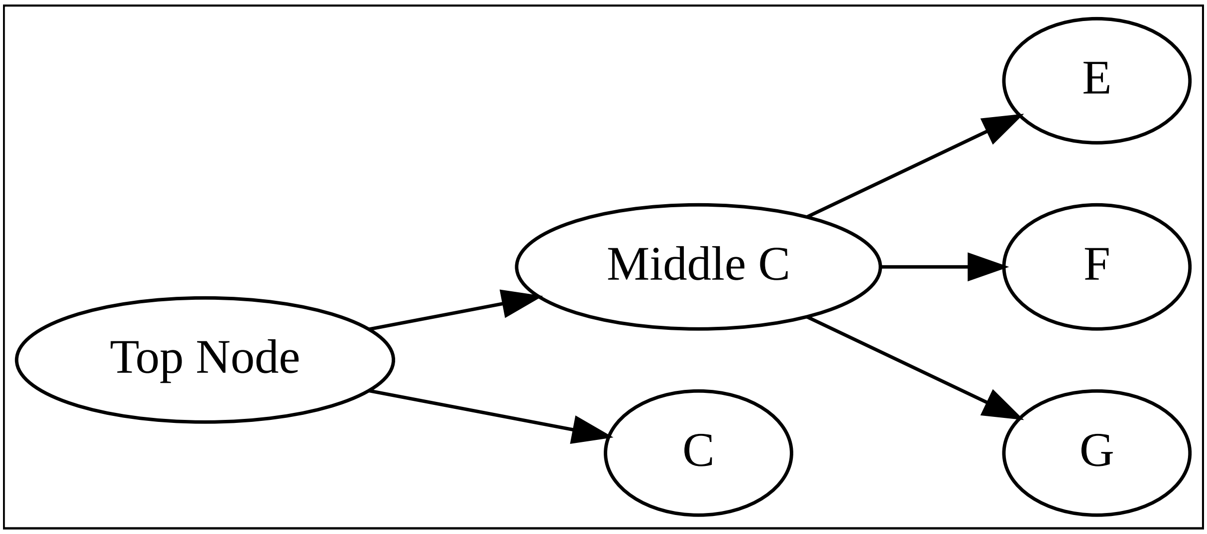

Figure 2. digraph