# Homework 4 - React

In this assignment, you will develop a React component that allows the reader of a web page to change the appearance of a piece of text.

When the user clicks on the text, a small form will appear that contains:

- a checkbox that lets the user choose whether the text should be bold
- buttons for decreasing and increasing the text's font size, as well as a display of the current size

Then, when the user clicks on the text again, the form disappears. Further details of the specification are provided below.

In completing this assignment, you will:

- Use the React framework to create a reusable, modular component
- Define callback functions that are invoked as the result of user actions in the HTML page
- Use those callback functions to modify the attributes of HTML elements

**Debugging/Error Note:**

If you run into errors/bugs/don't understand the output that Codio is giving you, please post in the Discussion Forum and a TA will assist you! Please do NOT email Codio as they will not review any errors you are getting.

**Getting Started**

Start by downloading the two files you will need for this assignment: right-click **this link** to save the file "chooser.html" to your computer; then right-click **this link** to save the file "FontChooser.js" to your computer.

FontChooser.js defines the FontChooser React component that you will implement in this assignment. Its render function uses JSX to return the following HTML elements within a `<div>`:

- a checkbox with the ID "boldCheckbox"
- a button with the ID "decreaseButton" and value "—"
- a span with the ID "fontSizeSpan" and value set to the React component's `this.props.size` value
- a button with the ID "increaseButton" and value "+"
- a span with the ID "textSpan" and value set to the React component's `this.props.text` value

Notice that the checkbox, two buttons, and “fontSizeSpan” are all initially hidden.

The chooser.html file is an HTML page that you can use for testing your React component. At the top, it includes `<script>` tags that include the three libraries required for React. You may use other React libraries if you choose, but we recommend these and will be using them for grading.

There is also a `<script>` tag that includes FontChooser.js, which will hold the definition of your React component, and is the file that you will submit for this assignment. Note that this line may cause an error when chooser.html is opened with Google Chrome. If so, then we recommend you use a different browser, e.g. Safari or Mozilla Firefox.

(It’s worth pointing out here that, ordinarily, we would want to create the React component in a separate .js file but this is not the way we would include it in the .html page. However, we will use this approach for now as it simplifies development and grading, and will see a better approach in later lessons this week.)

At the bottom of chooser.html is the `<div>` where the React component will be dropped, and then the call to `ReactDOM.render` that creates the FontChooser component with its different properties. Your implementation should, of course, work correctly with *any* specified properties, not just the ones shown here as an example.

When you open chooser.html in a browser such as Safari or Mozilla Firefox, you should see the text “Fun with React!” appear. If so, then you’re ready to start implementing this component.

**Activity**

The FontChooser component should allow the user to change the font weight (bold or normal) and font size of the text that it is displaying. Implement the React component in FontChooser.js as follows:

**Initial rendering**

- When the component is initially rendered, the checkbox, buttons, and “fontSizeSpan” element should be hidden, as in the version we distributed.
- The text that is set as the “text” property when the component is created should be displayed in the HTML page, as in the version we distributed.
- If the “bold” property is set to “true,” the text should be displayed in bold; otherwise it should be displayed as normal. The version we distributed does *not* include this functionality, so you will need to implement this.
- The text that is displayed should have a font size equal to the “size” property of the FontChooser component. The version we distributed does not include this functionality either.

**Displaying the form elements**

- When the checkbox, buttons, and “fontSizeSpan” element are hidden and the user clicks on the text that the component is displaying in the HTML page, the checkbox, buttons, and “fontSizeSpan” element should appear to the left of the text.
- When the checkbox, buttons, and “fontSizeSpan” element are shown and the user clicks on the text that the component is displaying in the HTML page, the checkbox, buttons, and “fontSizeSpan” element should disappear, i.e. become hidden again.

**Checkbox functionality**

- If the React component’s “bold” property is set to “true” when the component is initially created, the checkbox should be selected/checked when it is first displayed. If the “bold” property is set to “false,” the checkbox should be unselected/unchecked.

- If the checkbox is unselected/unchecked and then the user checks it, the text should immediately change to bold.
- If the checkbox is selected/checked and then the user unchecks it, the text should immediately change to normal font weight.

**Changing the font size**

- When the checkbox, buttons, and “fontSizeSpan” element are first displayed, the value in the “fontSizeSpan” should equal the React component’s “size” property, as in the version we distributed.
- When the user clicks the “decreaseButton” (the one with the “—” sign on it), the value in the “fontSizeSpan” should decrement and the font size of the text should immediately decrease by one as well. However, the value in the “fontSizeSpan” may not be smaller than the React component’s “min” property. If the value in the “fontSizeSpan” equals the “min” property and the user clicks the “decreaseButton,” there should be no change.
- Likewise, when the user clicks the “increaseButton” (the one with the “+” sign on it), the value in the “fontSizeSpan” should increment and the font size of the text should immediately increase by one as well. However, the value in the “fontSizeSpan” may not be larger than the React component’s “max” property. If the value in the “fontSizeSpan” equals the “max” property and the user clicks the “increaseButton,” there should be no change.
- If the value of the “fontSizeSpan” equals the component’s “min” or “max” property, then its color should be red. If “fontSizeSpan” is between “min” and “max,” though, then its color should be black. The font size of the text in the “fontSizeSpan” should always be the browser’s default, i.e. you do not need to explicitly set it.
- When the text in the “fontSizeSpan” is double-clicked, its value should become equal to the initial value set as the component’s initial “size” property, and the font size of the text that is displayed should immediately change to that value as well.

**Error handling and default values**

As you can see at the bottom of FontChooser.js, we have specified the default values for the props, in case they are not specified when the component is created.

However, your code should implement the following:

- If the “min” property has a value of 0 or a negative number, its value should be treated as 1 for limiting the smallest font size
- If the “min” property is greater than the “max” property, then “min” and “max” should both be treated as the larger of the two, i.e. the “min” property
- If the “size” property is less than the “min” property, the initial value should be treated as the same as the “min” property, or treated as 1 if “min” is 0 or negative
- If the “size” property is greater than the “max” property, the initial value should be treated as the same as the “max” property

You do not need to address any other combination of values not specified above; these are the only ones that will be considered for grading.

Additionally, you do not need to handle the situation in which the “text” property is not specified when the component is created.

**One more important note:**

Please do not change the ID attributes of any of the HTML elements defined in `FontChooser.render`, as these IDs will be used by our tests during grading. You are free to add other attributes to the HTML elements, such as “class” or “value” attributes, and you’ll of course need to register callback functions for various events, but do not change the IDs.

Likewise, please do not change the names of the properties that are set in the `<FontChooser>` tag in chooser.html so that the tests can use them for grading your submission.

**Helpful Hints**

Review the past few lessons to see the syntax for registering callback functions for events using React.

You may need to modify some HTML element attributes that were not explicitly addressed in the lessons, in particular:

- You can set whether an HTML element is displayed or not using its “hidden” attribute and setting it to “true” or “false” accordingly
- You can set whether the checkbox appears as selected/checked or unselected/unchecked using its “checked” attribute and setting it to “true” or “false” accordingly
- Your browser’s JavaScript console may show a warning about the checkbox having a “checked” attribute without an “onChange” handler. So be sure that when you modify the JSX for the component, the checkbox has a handler for “onChange” and not “onClick.”
- In React, use “onDoubleClick” as the event name to set up the callback handler for double-clicking on an element

Also, be sure to use “red” and “black” as the colors for the text in the “fontSizeSpan” and not, for instance, “RED” or its hexadecimal value or anything else, since the grader will be checking for “red” and “black” specifically. Likewise, the same goes for using “bold” and “normal” for the font weight.

**Before You Submit**

Please be sure that:

- You have not changed the “id” attribute of any of the HTML elements defined in `FontChooser.render`
- You have not created any additional files and all of your React code is in FontChooser.js

**Assessment**

Your submission will be assessed using automatic grading scripts that will check that the component works correctly for various inputs and events. Your score is determined by the percentage of these tests that “pass,” i.e. that produce the correct behavior for the specified input or event.

To submit your assignment, click the link below and follow the instructions in the Codio Readme file. You only need to upload **FontChooser.js** for grading.

## Homework #4 Submission (External resource) (100.0 points possible)

This link will take you to Codio so that you may submit this assignment.

**BEGIN SUBMISSION**