How To Guide Reference Document

# Traverse a Nested JSON in JavaScript

In this section, I’ll be walking through how to traverse a Nested JSON File. We won’t be focusing on the render of the data and we’ll just be printing out the value on the page.

## Theoretical Steps

- Create a function, this will allow it to be called within itself. Function will be talking in the JSON data

- Set up a loop to go through the JSON. It will take in the JSON and iterate through String Value Pair

- Now check if the value is either an Object, Array or just a primitive data type like numbers or strings

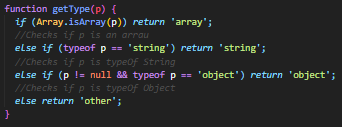
- If it’s an Object or an Array, print the key of the String Value Pair and called the function with the String Value Pair

- Else, print the String Value Pair and move onto the next pairing

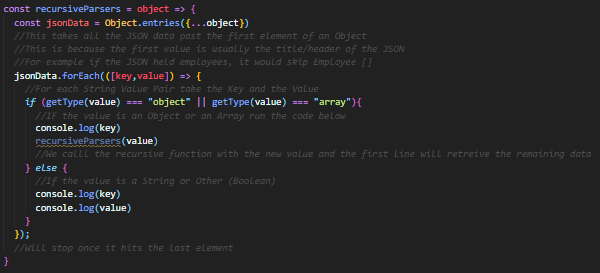
- The only time the Value of String Value Pair is printed is when the Value is identified as a String

## How to Detect Data Types

Below you can see a function I created to recognize if the data given is an Object, Array, String or other. I will be calling this through out the code to identify data types.



## The Recursive Parser

The code below is written in Javascript and take in pure JSON data without using libraries like Stringify. The comments below a line will be talking about the code above the comment. The purpose of this function, is to receive JSON data and print out all the String Value Pairs inside.

# Render a JSON using Recursion

This section will act as part 2 to the previous section on how to traverse a Nested JSON. I’ll be building upon previously mentioned code. So please make sure understand and have the code running.

Each time a complete String Value Pair that does not contain an object or array is found. A corresponding Label and Input Field will be created for said pair. If it’s an object or an array it will only print a Label but not an entry. This is because the value of the pair is an object or an array which doesn’t take in a value.

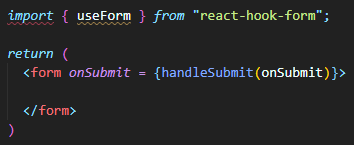
## React Hook Form

This is the library we’ll be using to obtain all the fields. To install run the command:

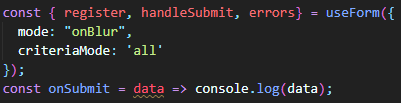
**npm i react-hook-form (**[react-hook-form - npm (npmjs.com)](https://www.npmjs.com/package/react-hook-form)).

## Adding React Rendering to Recursion

### Create a Form

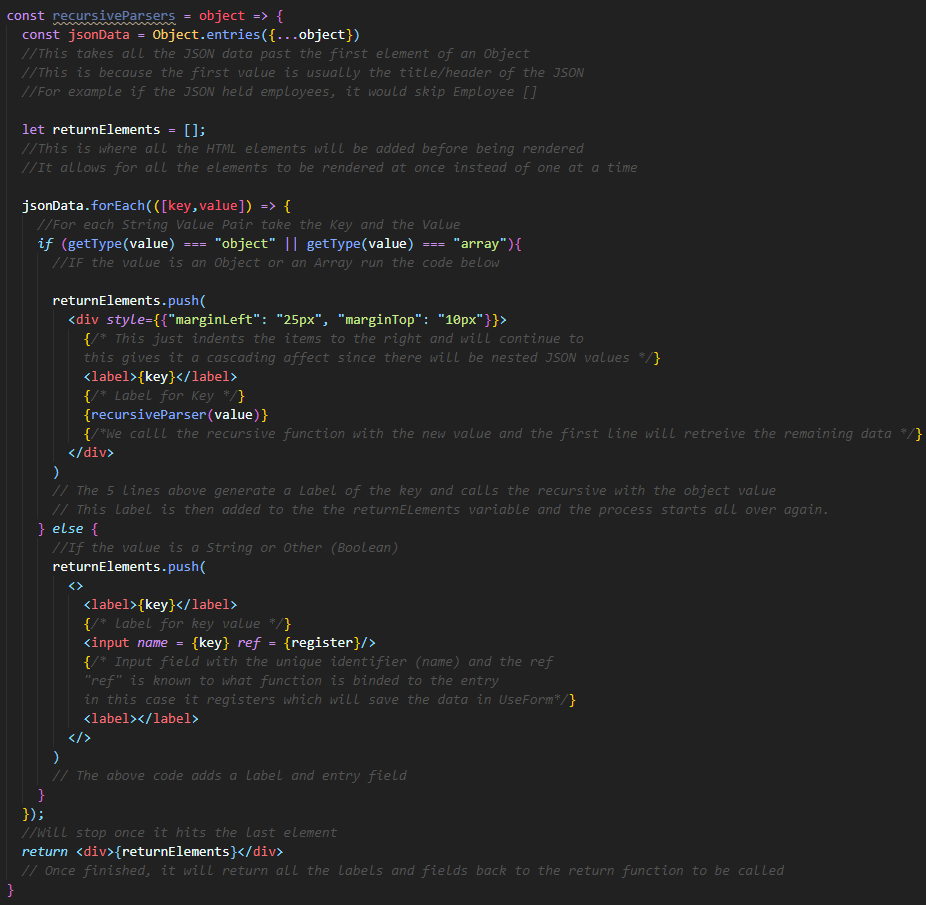


The attribute onSubmit refers to the function that will be called once the Submit button is pressed. Below you can see the code that is referred to. Once Submit is pressed all the data is printed in the console. Please add both the top and bottom code snippets to your code. (Ensure that the import statement is at the top above the class definition)



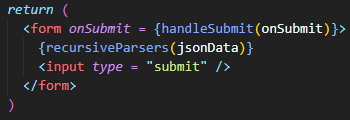
## Adjusting the Parser

This section we’ll be going through how to take the recursive parser from the previous section and start rendering the data through react. Below you can see the updates version of the recursive parser.



## Connecting the Recursive Function to the Form

You can now return all the rendered functions of the JSONs now you have to just connect it back to the Form you created on the first step. This is done simply by calling the function with the JSON data within the <Form></Form> like below. Also we add the submit button within the Form. This is all connected because we have used <Form>



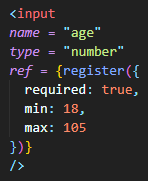
The reason everything is connected is because of useForm which connected all the <input> inside the <Form>. We may not be able to see actually Label and Input components, that is because it is coming from the return value of the recursiveParsers function.

# Implementing Real Time Validation

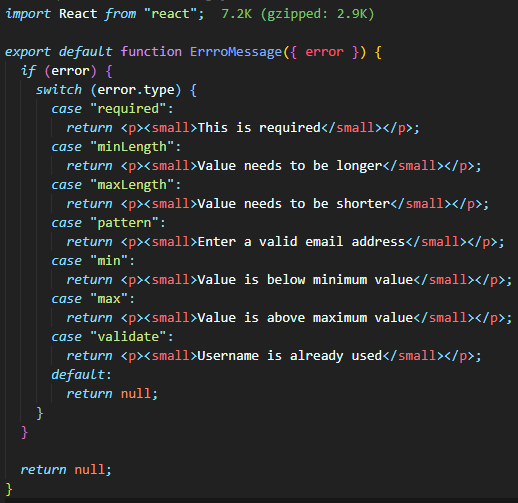
This validation is already built into the React Hook Form NPM module which is great. For this tutorial, it won’t be using all the possible validation methods the module has but it will be covering the basics. Please find the NPM Module Here.

## Declare Validation for a Field

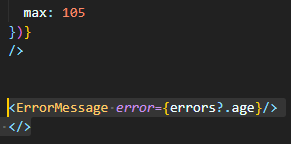
The code below, is defining an input that can only have numbers, has to be filled in, the minimum value is 18 and the maximum value is 105. As you can see the from the name parameter it’s age and that’s what it will be used for.



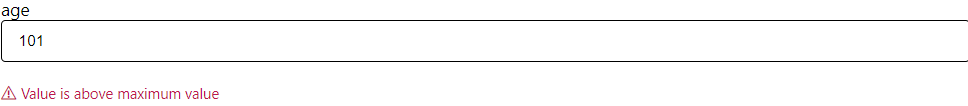
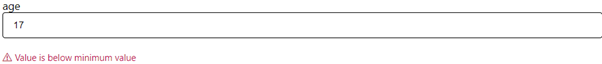
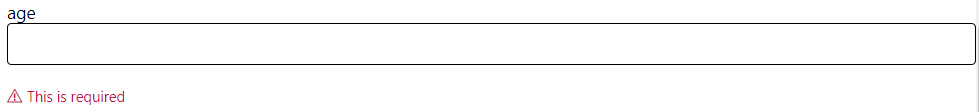
This is all well and good but nothing will appear when the input breaches one of the rules. We must declare another React component which will display the violation. Therefore you create an ErrorMessage component. Below is what the component looks like.



This is how it will be used with the Age input. You can see that the name parameter must be unique because that’s how the errormessage component connects to the input.



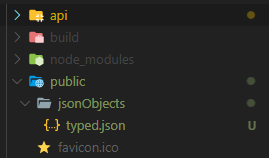
Now you have validation to the age input. This is what it looks like for each error!



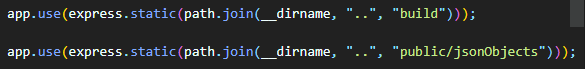
# How to Setup a Simple Backend (Express) and Connect to the Front End

This section is mainly based on connecting the backend to the frontend! This because I had followed a great tutorial on create a simple backend which can serve JSON files. ([How to Render a React App Using an Express Server in Node.js | by Yogesh Chavan | Level Up Coding (gitconnected.com)](https://levelup.gitconnected.com/how-to-render-react-app-using-express-server-in-node-js-a428ec4dfe2b))

## JSON Storage



For my code, my JSONs have been stored in the public folder inside the backend folder. I have added this specific pathway to my backend like this.

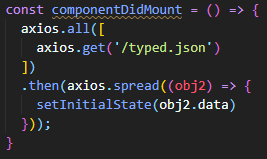


I tried my best to not use any absolute pathways to ensure that people can use my code easily.

## Query the Backend using the Frontend

My backend is hosted on port 3000. To query the backend manually to obtain I would enter <http://localhost:3000/filename.json>. This would display said file to the browser. For the frontend we just have to make a function that calls that port and reads the web content.

The function below is called within the same component as my recursive parser.



The function uses Axios which allows you to create HTTP requests within code. Specifically this function is calling for a file called “typed.json” from the path server/public/jsonObjects. Axios then reads all the data and sets it to a react state for storage. This state is then called by the JSON parser for rendering.

That’s really it, please follow the tutorial very thoroughly and ensure you do each step. It may seem a bit annoying to follow but trust me it works. If in doubt you can always pull my code from GitHub to see my own implementation.