JavaScript Department

Test Task

Gotoinc LTD

https://gotoinc.co/

Introduction

Dear **Candidate**,

Thank you very much for considering our company as the next great place for professional development and growth. We would really **appreciate** your time to complete this Test Task!

Please go through the tasks below and ask any questions before starting. Do your best to provide a preferable solution to the application according to its requirements.

Tasks

Implementing a Function to Group an Array of Objects by Property (with added complexity) - Up to 4 hours

Task Description:

Write a function `groupBy` that accepts three arguments:

1. **An array of objects** — this could be an array of any objects, each containing various properties.
2. **A property** or **a grouping function** — this can be:
   - The name of a property to group the elements by.
   - A function that takes an object and returns a key for grouping.

The function should return an object where the keys are the values of the property or the result of the function, and the values are arrays of objects sharing the same key.

Requirements:

1. **Support for nested grouping**: Implement the ability to pass an array of properties or functions for grouping at multiple levels of nesting.
   For example, first group objects by country, and then group by city within each group.
2. **Handling missing or incorrect properties**: If a property is missing from some objects or the function returns `undefined`, such elements should be grouped under the key `"undefined"`.
3. **Ability to work with arrays of different data types**: Objects can contain different data types (strings, numbers, date, etc).

4. **Filtering parameter**: Add the ability to pass an additional function to filter elements before grouping. For example, group only those objects where the value of a specific property exceeds a certain threshold.

Sample of the initial data and output you can find **here**.

Creating a Layout Using HTML, CSS, and SCSS - Up to 6 hours

Task Description:

Develop a responsive webpage section based on a given design using HTML for structure, CSS/SCSS for styling, and ensure adherence to modern best practices in front-end development. You will be provided with a design layout in Figma with detailed requirements, and you need to create a responsive web page according to that design.

Requirements:

1. **HTML**:
    - Use semantic tags to structure the content.
    - Ensure correct markup for section.
2. **CSS/SCSS**:
    - Use of SCSS preprocessor to create variables, mixins, and nested styles.
    - No use of third-party CSS frameworks (Bootstrap, Tailwind, etc.) — all code must be written from scratch.
    - Organize SCSS structure using any CSS methodology.
    - Ensure the page is responsive screen sizes (360px and 1440px from design). The layout should look correct on both mobile devices and desktops due to the design.
    - Add all effects for interactive elements.
    - Provide minimal animation for enhanced UX (e.g., element fade-in or smooth transitions).
3. **Will be a plus (not required):**
    - Implement full version of the page using js (could be done just a few from the list below):
        1. Implement several breakpoints for smooth transitions between mobile, tablet, and desktop versions.
        2. Implement preview of the main select image.
        3. Implement switch of size type.
        4. Implement tabs for product details.
        5. Implement select size type.
        6. Implement “Size guide” dialog

The figma designs and prototypes you can find by following links.

Implementation of a Next.js Posts App - Up to 8 hours

Task Description:

Build a Next.js app that includes a login form (client-side and server-side), protected routes, a list of posts from the JSONPlaceholder API, a post details page, and dynamic metadata configuration for the post pages.

Requirements:

1. **Login Form (Server/Client)**:
   - Create a login form component.
   - Fields: `username`, `password`.
   - Hardcode credentials for the form (e.g., `username: "admin"`, `password: "1234"`).
   - After entering the correct credentials, make a request to a Next.js API (e.g., `/api/login`) to validate the login.
   - If login is successful, store a token (or session) in localStorage or cookies.
   - Display an error message if login fails.
2. **Protected Routes**:
   - Use middleware to protect specific pages (e.g., `/dashboard`).
   - Check for a valid token or session before rendering the page.
   - If the user is not authenticated, redirect them to the login page.
3. **Homepage with Posts**:
   - On the homepage (`/`), display a list of posts fetched from the JSONPlaceholder API.
   - Use server-side data fetch in React Server Components.
   - Display the title and a short description for each post.
   - Add functionality to click on a post, which navigates to the post's details page.
4. **Single Post Page**:
   - When a post is clicked, the user will be taken to the post details page (route format: `/posts/[id]`).
   - Use server-side data fetch in React Server Components for fetching the post data.
   - Display the full details of the post (post title, description, and body) on this page.
5. **Metadata Configuration for Single Post Page**:
   - Configure dynamic metadata using Metadata API for each individual post page using the fetched post data.
   - Update the metadata fields dynamically, such as the `<title>`, `<meta description>`, and other SEO-related attributes based on the post's information.

Deadline

From the moment you receive the task, you will have one week (7 days) to complete the task.

Publication

When you're done, but no later than the deadline, send a link to the project that should be posted to your GitHub account.

The Gotoinc team wishes you success in your task!