The function **productDetails**(a, b) is designed to validate input and then return formatted details for a specific product from an array of objects.

### Function Purpose

The function takes two arguments:

* a: An array of product objects.
* b: A number representing the id of a specific product within the array.

The function validates the inputs and returns formatted information about the product with the given id.

**Validation Steps**

1. **Check the number of arguments:**
   * The function expects exactly two arguments. If the number of arguments is not 2, it returns an error message: 'Ši funkcija reikalauja įvesti tiksliai 2 kintamuosius', which means "This function requires exactly 2 variables" in Lithuanian.
2. **Validate that a is an array:**
   * If a is not an array, it returns 'Pirmoji reikšmė turi būti masyvas' ("The first value must be an array").
3. **Check that the array a is not empty:**
   * If a is an empty array, it returns 'Pirmoji reikšmė, masyvas turi turėti objektą' ("The first value, the array, must contain objects").
4. **Ensure all elements in a are non-empty objects:**
   * If any element in a is not an object, is null, is an array, or is an empty object, it returns 'Pirmoji reikšmė - masyve turi būti tik objektai ir jie turi būti ne tušti' ("The first value - the array must contain only objects and they must not be empty").
5. **Validate that b is a finite number:**
   * If b is not a finite number, it returns 'Antroji reikšmė turi būti skaičius' ("The second value must be a number").
6. **Check if a product with the given id exists in the array:**
   * If no object in a has an id matching b, it returns 'Prekė, su ID: ${b} neegzistuoja.' ("Product with ID: ${b} does not exist").

### Processing and Output

If all validations pass:

1. The function locates the product object in the array a whose id matches b.
2. It then calls an inner function formatProductDetails(product) that:
   * Constructs a formatted string of the product’s details including:
     + ID
     + Name (Pavadinimas)
     + Amount (Kiekis)
     + Unit price (Vieneto kaina)
     + Total price (Viso mokėti)
   * Aligns these details neatly using headers and dynamically calculated padding.
3. The formatted product details string is returned.

**Example Input and Output**

**Input:**

const products = [

{ id: 1, name: 'Apple', amount: 5, unitPrice: 100 },

{ id: 2, name: 'Banana', amount: 3, unitPrice: 150 },

];

productDetails(products, 1);

**Output:**

-------------------------

Prekės informacija

-------------------------

ID | 1

Pavadinimas | Apple

Kiekis | 5 vnt

Vieneto kaina | 1.00 Eur

Viso mokėti | 5.00 Eur

-------------------------

### Conclusion

This function is useful for generating a formatted summary of a product's details based on its id from an array of product objects, with robust validation to ensure correct input types and values.

The **shoppingList**(a) function generates a formatted shopping list from an array of items, ensuring the input is valid and producing a readable output that details each item's name, quantity, unit price, and total price. Here's how the function works:

### Function Purpose

The function takes a single argument a, which is expected to be an array of objects representing items in a shopping cart. Each object should have the properties:

* name: Name of the product.
* amount: Quantity of the product.
* unitPrice: Price per unit (in cents, not euros).

### Validation Steps

1. **Check the number of arguments:**
   * The function requires exactly one argument. If the function is called with more or fewer arguments, it returns an error message: 'Ši funkcija reikalauja įvesti tiksliai 1 kintamajį', which means "This function requires exactly 1 variable" in Lithuanian.
2. **Validate that a is an array:**
   * If a is not an array, the function returns 'Kintamasis turi būti masyvas' ("The variable must be an array").
3. **Check that the array a is not empty:**
   * If a is an empty array, the function returns 'Šiuo metu, jūsų prekių krepšelis yra tuščias.' ("Currently, your shopping cart is empty").

### Processing and Output

If the input passes all validations, the function proceeds to generate a formatted shopping list:

1. **Calculate the number of items:**
   * The function determines how many items are in the cart (totalItems) and uses the correct word for "item" in Lithuanian (prekė for singular, prekės for plural).
2. **Prepare the intro and header lines:**
   * An introductory line summarizes the total number of items.
   * A header line outlines the columns for the product details (Pavadinimas, Kiekis, Vieneto kaina, and Viso mokėti).
3. **Format each item’s details:**
   * The formatShoppingList(list) function maps over each item in a, creating a formatted string with its:
     + Index number in the list.
     + Name.
     + Quantity with "vnt" (units).
     + Unit price in euros (converted from cents and rounded to 2 decimal places).
     + Total price (quantity multiplied by unit price, also in euros).
   * The resulting formatted strings are stored in formattedLines.
4. **Construct the final output:**
   * The function combines the intro line, header line, item lines, and separator lines (hyphenLine) into a complete string finalTableOutput.
   * The string is trimmed of extra spaces and returned.

### Example Input and Output

**Input:**

const items = [

{ name: 'Apple', amount: 4, unitPrice: 120 },

{ name: 'Banana', amount: 2, unitPrice: 150 },

];

shoppingList(items);

**Output:**

Jūsų prekių krepšelyje yra 2 prekės:

-------------------------------------------------

Pavadinimas | Kiekis | Vieneto kaina | Viso mokėti

-------------------------------------------------

1. Apple | 4 vnt | 1.20 Eur | 4.80 Eur

2. Banana | 2 vnt | 1.50 Eur | 3.00 Eur

-------------------------------------------------

### Conclusion

The shoppingList(a) function is designed to neatly format and present a list of shopping cart items. It handles validation to ensure the input is correct, and produces a structured and human-readable output detailing the contents of the shopping cart, including quantities, unit prices, and total costs.

INDEX file is a JavaScript script that demonstrates the use of the **shoppingList** and **productDetails** functions by importing them from their respective modules and running a series of test cases.

Imports

import { shoppingList } from "./js/shoppingList.js";

import { productDetails } from "./js/productDetails.js";

Testing an Empty Shopping List:

const emptyList = [];

console.log(shoppingList(emptyList));

console.log(productDetails(emptyList, 42069));

Testing a Full Shopping List:

const firstShoppingList = [

{ id: 1, name: 'Pomidoras', amount: 1000000, unitPrice: 199 },

{ id: 3, name: 'Agurkas', amount: 2, unitPrice: 50 },

{ id: 7, name: 'Svogūnas', amount: 1, unitPrice: 45 },

];

console.log(shoppingList(firstShoppingList));

console.log(productDetails(firstShoppingList, 42069));

console.log(productDetails(firstShoppingList, 1));

console.log(productDetails(firstShoppingList, 3));

console.log(productDetails(firstShoppingList, 7));

Testing a Single Product Shopping List:

const singleProductShoppingList = [

{ id: 2, name: 'Kivi', amount: 23, unitPrice: 14 },

];

console.log(shoppingList(singleProductShoppingList));

console.log(productDetails(singleProductShoppingList, 42069));

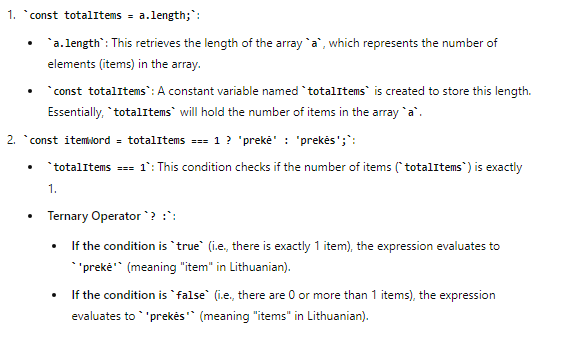
console.log(productDetails(singleProductShoppingList, 2));

kaip nustatoma prekes ar preke

const totalItems = a.length;

const itemWord = totalItems === 1 ? 'prekė' : 'prekės';

These lines of code in JavaScript calculate the total number of items in an array and determine the correct form of a word based on that number.



**Summary:**

* If the array a has exactly 1 item, itemWord will be 'prekė'.
* If the array a has 0 or more than 1 item, itemWord will be 'prekės'.

This is a common way to handle pluralization of words based on quantity.