Resurse suplimentare
pe care recomand să le parcurgi

Partea 1: Calculator de Statistici

În această parte, vei crea un calculator de statistici care utilizează funcții pentru a procesa un array de numere.

Cerințe:

1. Creează un array cu minim 10 numere.
2. Implementează următoarele funcții:
    - `calculateAverage(numbers)` - returnează media numerelor
    - `findMax(numbers)` - returnează cel mai mare număr
    - `findMin(numbers)` - returnează cel mai mic număr
    - `calculateSum(numbers)` - returnează suma numerelor
3. Apelează funcțiile și afișează rezultatele în consolă.

Cod de pornire:

```
// Array de numere pentru analiză
const numbers = [12, 7, 19, 23, 8, 10, 17, 22, 14, 9]

// Implementează funcțiile de statistică

// Funcția pentru calcularea mediei
function calculateAverage(numbers) {
  // Implementează codul aici
}

// Funcția pentru găsirea maximului
function findMax(numbers) {
  // Implementează codul aici
}

// Funcția pentru găsirea minimului
function findMin(numbers) {
  // Implementează codul aici
}

// Funcția pentru calcularea sumei
function calculateSum(numbers) {
  // Implementează codul aici
}

// Afișează rezultatele
console.log('Statistici pentru array-ul:', numbers)
console.log('Media:', calculateAverage(numbers))
console.log('Maximul:', findMax(numbers))
console.log('Minimul:', findMin(numbers))
console.log('Suma:', calculateSum(numbers))
```

Extindere:

Adaugă o funcție suplimentară `filterEven(numbers)` care returnează un nou array conținând doar numerele pare.

Partea 2: Gestiune de Inventar

Creează un sistem simplu de gestionare a inventarului folosind obiecte și array-uri.

Cerințe:

1. Creează un array de obiecte, fiecare reprezentând un produs cu proprietățile: id, name, price, quantity.
2. Implementează următoarele funcții:
    - `addProduct(inventory, product)` - adaugă un produs nou în inventar
    - `findProduct(inventory, id)` - găsește un produs după ID
    - `updateQuantity(inventory, id, newQuantity)` - actualizează cantitatea unui produs
    - `calculateTotalValue(inventory)` - calculează valoarea totală a inventarului (sumă de price * quantity)

Cod de pornire:

```javascript
// Inventarul inițial de produse
const inventory = [
  { id: 1, name: 'Laptop', price: 2500, quantity: 5 },
  { id: 2, name: 'Telefon', price: 1200, quantity: 10 },
  { id: 3, name: 'Tabletă', price: 800, quantity: 8 },
]

// Implementează funcțiile de gestiune a inventarului

function addProduct(inventory, product) {
  // Implementează codul aici
}

function findProduct(inventory, id) {
  // Implementează codul aici
}

function updateQuantity(inventory, id, newQuantity) {
  // Implementează codul aici
}

function calculateTotalValue(inventory) {
  // Implementează codul aici
}

// Testare funcții
console.log('Inventarul inițial:', inventory)

// Adaugă un produs nou
const newProduct = { id: 4, name: 'Monitor', price: 700, quantity: 3 }
addProduct(inventory, newProduct)
console.log('Inventar după adăugare:', inventory)

// Găsește un produs
console.log('Produsul cu ID 2:', findProduct(inventory, 2))

// Actualizează cantitatea
updateQuantity(inventory, 3, 12)
console.log('Inventar după actualizare:', inventory)

// Calculează valoarea totală
console.log('Valoarea totală a inventarului:', calculateTotalValue(inventory))
```

Extindere:

Adaugă funcționalitatea de a șterge un produs din inventar folosind funcția `deleteProduct(inventory, id)`.

Partea 3: Transformare de Date cu Array Methods

Creează un program care utilizează metodele avansate de array pentru a transforma și filtra date.

Cerințe:

1. Ai un array de obiecte reprezentând studenți.
2. Folosind metode de array precum map, filter, reduce, și sort, realizează următoarele operații:
    - Filtrează studenții care au note de trecere (peste 5)
    - Sortează studenții după medie, în ordine descrescătoare
    - Transformă array-ul pentru a include doar numele și media fiecărui student
    - Calculează media generală a tuturor studenților

Cod de pornire:

```
// Date despre studenți
const students = [
  { name: 'Ana Ionescu', age: 21, average: 9.5 },
  { name: 'Mihai Popescu', age: 22, average: 8.3 },
  { name: 'Elena Dumitrescu', age: 20, average: 7.8 },
  { name: 'Andrei Stanescu', age: 23, average: 6.4 },
  { name: 'Maria Constantinescu', age: 21, average: 9.1 },
  { name: 'Ion Vasilescu', age: 22, average: 5.2 },
  { name: 'Ioana Munteanu', age: 20, average: 4.9 },
]

// 1. Filtrează studenții care au note de trecere (peste 5)
const passingStudents = null // Implementează folosind filter()

// 2. Sortează studenții după medie, în ordine descrescătoare
const sortedStudents = null // Implementează folosind sort()

// 3. Transformă array-ul pentru a include doar numele și media
const nameAndAverage = null // Implementează folosind map()

// 4. Calculează media generală a tuturor studenților
const overallAverage = null // Implementează folosind reduce()

// Afișarea rezultatelor
console.log('Studenți cu note de trecere:', passingStudents)
console.log('Studenți sortați după medie:', sortedStudents)
console.log('Nume și medie:', nameAndAverage)
console.log('Media generală:', overallAverage)
```

Partea 4: Gestionarea Erorilor în Aplicații

Dezvoltă o mini-aplicație de validare a datelor cu gestionare corespunzătoare a erorilor.

Cerințe:

1. Creează o funcție `validateUser(user)` care verifică dacă un obiect utilizator este valid.

2. Verifică următoarele reguli folosind gestionarea erorilor:
    - Numele trebuie să aibă cel puțin 3 caractere
    - Email-ul trebuie să conțină caracterul '@'
    - Vârsta trebuie să fie un număr între 18 și 120
3. Aruncă erori personalizate pentru fiecare condiție neîndeplinită.
4. Tratează erorile folosind un bloc try-catch.

Cod de pornire:

```
// Clasa personalizată pentru erori de validare
class ValidationError extends Error {
  constructor(message, field) {
    super(message)
    this.name = 'ValidationError'
    this.field = field
  }
}

// Funcția de validare
function validateUser(user) {
  // Implementează validările aici și aruncă erori corespunzătoare
}

// Teste pentru funcția de validare
const users = [
  { name: 'Ana Popescu', email: 'ana@example.com', age: 25 },
  { name: 'Io', email: 'ion-example.com', age: 17 },
  { name: 'Maria Ionescu', email: 'maria@example.com', age: 130 },
  { name: 'Gheorghe Popa', email: 'gheorghe@example.com', age: 'treizeci' },
]

// Verifică fiecare utilizator
users.forEach((user, index) => {
  console.log(`\\nVerificare utilizator ${index + 1}:`, user)
  try {
    validateUser(user)
    console.log('✅ Utilizator valid!')
  } catch (error) {
    if (error instanceof ValidationError) {
      console.log(
        `❌ Eroare de validare pentru câmpul ${error.field}: ${error.message}`
      )
    } else {
      console.log(`❌ Eroare neașteptată: ${error.message}`)
    }
  }
})
```

Partea 5: Proiect - Aplicație de Gestionare a Sarcinilor

Creează o mini-aplicație pentru gestionarea sarcinilor (to-do list) care să folosească toate conceptele învățate.

Cerințe:

1. Creează un obiect `taskManager` care să aibă următoarele funcționalități:
    - Adăugarea unei sarcini noi (cu proprietățile: id, title, description, priority, completed)
    - Marcarea unei sarcini ca terminată
    - Ștergerea unei sarcini
    - Filtrarea sarcinilor după stare (completed/uncompleted)
    - Sortarea sarcinilor după prioritate
    - Calcularea procentajului de sarcini finalizate
2. Implementează gestionarea erorilor pentru cazuri precum:
    - ID duplicat
    - Sarcină inexistentă
    - Date invalide

Cod de pornire:

```javascript
// Manager de sarcini
const taskManager = {
  tasks: [],

  // Adaugă o sarcină nouă
  addTask(title, description, priority) {
    // Implementează codul aici
  },

  // Marchează o sarcină ca finalizată
  completeTask(id) {
    // Implementează codul aici
  },

  // Șterge o sarcină
  deleteTask(id) {
    // Implementează codul aici
  },

  // Filtrează sarcinile după starea lor
  filterTasks(completed = true) {
    // Implementează codul aici
  },

  // Sortează sarcinile după prioritate (1 = mare, 3 = mică)
  sortByPriority() {
    // Implementează codul aici
  },

  // Calculează procentul de finalizare
  completionPercentage() {
    // Implementează codul aici
  },
}

// Testare funcţionalităţi
try {
  // Adăugare sarcini
  taskManager.addTask(
    'Învaţă JavaScript',
    'Studiază funcţiile, array-urile şi obiectele',
    1
  )
  taskManager.addTask('Cumpărături', 'Lapte, pâine, ouă', 2)
  taskManager.addTask('Plimbare', '30 minute în parc', 3)

  console.log('Sarcini iniţiale:', taskManager.tasks)

  // Finalizare sarcină
  taskManager.completeTask(2)
  console.log('După finalizare:', taskManager.tasks)

  // Sortare după prioritate
  const sortedTasks = taskManager.sortByPriority()
  console.log('Sarcini sortate după prioritate:', sortedTasks)

  // Filtrare sarcini finalizate
  const completedTasks = taskManager.filterTasks(true)
  console.log('Sarcini finalizate:', completedTasks)

  // Procentaj finalizare
  console.log(
    'Procent de finalizare:',
    taskManager.completionPercentage() + '%'
  )
} catch (error) {
  console.error('A apărut o eroare:', error.message)
}
```