Exercice 1:

function Voiture(model, marque, annee, type, carburant) {

this.model = model;

this.marque = marque;

this.annee = annee;

this.type = type;

this.carburant = carburant;

}

let voitures = [

new Voiture("Accent", "Hyundai", 2020, "Berline", "Essence"),

new Voiture("Fiesta", "Ford", 2018, "Compacte", "Diesel"),

new Voiture("Tucson", "Hyundai", 2021, "SUV", "Hybride"),

new Voiture("Focus", "Ford", 2019, "Berline", "Essence")

];

function Hyundai(model, marque, annee, type, carburant, serie, hybride) {

Voiture.call(this, model, marque, annee, type, carburant);

this.serie = serie;

this.hybride = hybride;

}

Hyundai.prototype = Object.create(Voiture.prototype);

Hyundai.prototype.constructor = Hyundai;

Hyundai.prototype.alarmer = function() {

return "Alarme de " + this.marque + " " + this.model + " activée!";

};

function Ford(model, marque, annee, type, carburant, options) {

Voiture.call(this, model, marque, annee, type, carburant)

this.options = options;

}

Ford.prototype = Object.create(Voiture.prototype);

Ford.prototype.constructor = Ford;

voitures.sort(function(a, b) {

return a.annee - b.annee;

});

console.log("Voitures triées par année:");

voitures.forEach(function(voiture) {

console.log(voiture.marque + " " + voiture.model + " (" + voiture.annee + ")");

});

Excusion:



Exercice2:

function Etudiant(nom, prenom, age, cne) {

this.nom = nom;

this.prenom = prenom;

this.age = age;

this.cne = cne;

}

function Professeur(nom, age, cin) {

this.nom = nom;

this.age = age;

this.cin = cin;

}

Etudiant.prototype.etudier = function() {

return this.nom + " " + this.prenom + " est en train d'étudier.";

};

Professeur.prototype.enseigner = function() {

return "Professeur " + this.nom + " est en train d'enseigner.";

};

let etudiants = [

new Etudiant("Dupont", "Jean", 20, "CNE123"),

new Etudiant("Martin", "Sophie", 19, "CNE456"),

new Etudiant("Durand", "Pierre", 21, "CNE789"),

new Etudiant("Dupont", "Marie", 20, "CNE101")

];

etudiants.sort(function(a, b) { if (a.nom < b.nom) return -1;

if (a.nom > b.nom) return 1;

if (a.prenom < b.prenom) return -1;

if (a.prenom > b.prenom) return 1;

return a.age - b.age;

});

console.log("Étudiants triés par nom, prénom et âge:");

etudiants.forEach(function(etudiant) {

console.log(etudiant.nom + " " + etudiant.prenom + ", " + etudiant.age + " ans");

});

Exucion :



Exercice3:

class Vecteur2D {

constructor(x = 0, y = 0) {

this.x = x;

this.y = y;

}

afficher() {

return `Vecteur(${this.x}, ${this.y})`;

}

add(autre) {

return new Vecteur2D(this.x + autre.x, this.y + autre.y);

}

}

console.log("Test de la classe Vecteur2D:");

const v1 = new Vecteur2D();

const v2 = new Vecteur2D(3, 4);

console.log(v1.afficher()); // Affiche: Vecteur(0, 0)

console.log(v2.afficher());

Excusion:



Exercice 4:

class User {

constructor(id, username, email) {

this.id = id;

this.username = username;

this.email = email;

} toJSON() {

return {

id: this.id,

username: this.username,

email: this.email

};

}

static fromJSON(json) {

return new User(json.id, json.username, json.email);

}

}

class Post {

constructor(id, title, content, userId, date = new Date()) {

this.id = id;

this.title = title;

this.content = content;

this.userId = userId;

this.date = date;

}

toJSON() {

return {

id: this.id,

title: this.title,

content: this.content,

userId: this.userId,

date: this.date.toISOString()

};

}

static fromJSON(json) {

return new Post(

json.id,

json.title,

json.content,

json.userId,

new Date(json.date)

);

}

}

class Blog {

constructor() {

this.users = [];

this.posts = [];

this.nextUserId = 1;

this.nextPostId = 1;

}

addUser(username, email) {

const user = new User(this.nextUserId++, username, email);

this.users.push(user);

return user;

}

addPost(title, content, userId) {

const post = new Post(this.nextPostId++, title, content, userId);

this.posts.push(post);

return post;

}

getAllPosts() {

return this.posts;

}

getPostsByUser(userId) {

return this.posts.filter(post => post.userId === userId);

}

toJSON() {

return {

users: this.users.map(user => user.toJSON()),

posts: this.posts.map(post => post.toJSON()),

nextUserId: this.nextUserId,

nextPostId: this.nextPostId

};

}

static fromJSON(json) {

const blog = new Blog();

blog.nextUserId = json.nextUserId;

blog.nextPostId = json.nextPostId;

blog.users = json.users.map(User.fromJSON);

blog.posts = json.posts.map(Post.fromJSON);

return blog;

}

saveToFile(fs, filename) {

const json = JSON.stringify(this.toJSON(), null, 2);

fs.writeFileSync(filename, json);

console.log(`Blog sauvegardé dans ${filename}`);

}

static loadFromFile(fs, filename) {

try {

const json = JSON.parse(fs.readFileSync(filename, 'utf8'));

return Blog.fromJSON(json);

} catch (error) {

console.error(`Erreur lors du chargement depuis ${filename}:`, error.message);

return new Blog(); // Retourner un nouveau blog vide en cas d'erreur

}

}

}

const blog = new Blog();

const user1 = blog.addUser("john\_doe", "john@example.com");

const user2 = blog.addUser("jane\_smith", "jane@example.com");

blog.addPost("Premier article", "Contenu du premier article", user1.id);

blog.addPost("Deuxième article", "Contenu du deuxième article", user1.id);

blog.addPost("Article de Jane", "Contenu de l'article de Jane", user2.id);

console.log("Tous les posts:");

blog.getAllPosts().forEach(post => {

console.log(`${post.title} (par utilisateur #${post.userId})`);

});

console.log("\nPosts de l'utilisateur #1:");

blog.getPostsByUser(user1.id).forEach(post => {

console.log(`${post.title}: ${post.content}`);

});

console.log("\nDémonstration de conversion en JSON:");

const blogJSON = JSON.stringify(blog.toJSON(), null, 2);

console.log(blogJSON);

const fs = require('fs');

blog.saveToFile(fs, 'blog-ata.json');

const loadedBlog = Blog.loadFromFile(fs, 'blog-data.json');

console.log("\nBlog chargé depuis le fichier:");

console.log(`Nombre d'utilisateurs: ${loadedBlog.users.length}`);

console.log(`Nombre de posts: ${loadedBlog.posts.length}`);

Exercice 5:

console.log("1. GESTION DES LIVRES (ARRAY):");

let books = ["Le Petit Prince", "1984", "Harry Potter"];

console.log("Tableau initial:", books);

books.push("Don Quichotte");

console.log("Après push:", books);

books.unshift("Les Misérables");

console.log("Après unshift:", books);

let removedLastBook = books.pop();

console.log("Livre retiré avec pop:", removedLastBook);

console.log("Après pop:", books);

let removedFirstBook = books.shift();

console.log("Livre retiré avec shift:", removedFirstBook);

console.log("Après shift:", books);

console.log("\n2. GESTION DES CATÉGORIES (SET):");

let categories = new Set(["Fiction", "Science", "Histoire"]);

console.log("Set initial:", categories);

categories.add("Philosophie");

console.log("Après ajout de 'Philosophie':", categories);

categories.add("Fiction");

console.log("Après tentative d'ajout de 'Fiction' (déjà existant):", categories);

categories.delete("Science");

console.log("Après suppression de 'Science':", categories);

console.log("\n3. GESTION DES EMPRUNTS (MAP):");

let borrows = new Map();

borrows.set("Le Petit Prince", "Alice");

borrows.set("1984", "Bob");

borrows.set("Don Quichotte", "Charlie");

console.log("Map initial:");

borrows.forEach((emprunteur, livre) => {

console.log(`- "${livre}" emprunté par ${emprunteur}`);

});

borrows.delete("1984");

console.log("\nAprès suppression de '1984':");

borrows.forEach((emprunteur, livre) => {

console.log(`- "${livre}" emprunté par ${emprunteur}`);

});

let bookToCheck = "Le Petit Prince";

console.log(`\nVérification si "${bookToCheck}" est emprunté:`, borrows.has(bookToCheck));

bookToCheck = "Harry Potter";

console.log(`Vérification si "${bookToCheck}" est emprunté:`, borrows.has(bookToCheck));