UNIVERSITATEA DE STAT DIN MOLDOVA

Facultatea de Matematică și Informatică

Departamentul de Informatică

CURRICULUM

la unitatea de curs

JavaScript

Ciclul I, Licență

Program / Specialitatea: 0613.5 Informatică Aplicată

AUTORI:

Nartea Nichita, asis. univ.

Donu Alexandru, asis. univ.

APROBAT
la şedinţa Departamentului
din _____ _____ 2025
proces-verbal nr. _____
Şef Departament _____

APROBAT
la şedinţa Consiliului Facultăţii
din _____ _____ 2025
proces-verbal nr. _____
Şef Departament _____

Chișinău, 2025

Conținut

PRELIMINARII

Prezentarea generală a cursului

În contextul evoluției accelerate a tehnologiilor digitale și a cererii tot mai mari de specialiști IT, limbajul *JavaScript* a devenit o competență esențială pentru orice dezvoltator. Deși este cunoscut în special pentru rolul său în crearea aplicațiilor web interactive și dinamice, *JavaScript* este utilizat pe scară largă și în alte domenii precum dezvoltarea aplicațiilor mobile și desktop, DevOps, securitatea cibernetică, automatizări și chiar inteligența artificială.

Cursul *JavaScript* le va oferi studenților o bază solidă în acest limbaj versatil, pornind de la elementele fundamentale — structura codului, variabile, funcții — și continuând cu manipularea DOM, gestionarea evenimentelor și programarea asincronă. În plus, vor fi abordate concepte avansate precum callback-uri, promises și async/await, pregătind astfel studenții pentru o gamă largă de aplicații reale în industrie.

Scopul principal al cursului

Scopul principal al cursului este de a echipa studenții cu cunoștințele și abilitățile necesare pentru a dezvolta aplicații web eficiente, folosind *JavaScript*. De asemenea, cursul își propune să învețe studenții cum să scrie cod curat și modular, aplicând principii de dezvoltare moderne și bune practici de programare.

Locul și rolul cursului în formarea rezultatelor învățării ale specialității și misiunea curriculumului în formarea profesională

Cursul *JavaScript* contribuie esențial la formarea competențelor fundamentale necesare în domeniul dezvoltării web și tehnologiilor informaționale. Studenții vor dobândi:

- O înțelegere solidă a principiilor de programare în JavaScript;
- Abilități de manipulare a DOM și de gestionare a interacțiunilor utilizator-aplicație;
- Capacitatea de a scrie cod modular și reutilizabil;
- O bază solidă pentru a continua studiile în domeniul programării și al dezvoltării web.

Misiunea acestui curs este de a forma specialiști capabili să dezvolte soluții web moderne și eficiente, aplicând JavaScript în mod corect și utilizând cele mai bune practici din industrie.

Limba de predare a cursului

Instruirea se va desfășura în limba română și rusă.

Beneficiarii

Beneficiarii acestui curs sunt studenții de la anul 1 licență, specialitatea Informatică Aplicată și Informatica. De asemenea, cursul poate fi util și altor studenți sau profesioniști interesați.

I. ADMINISTRAREA DISCIPLINEI

Forma de învățământ	Codul unității de curs	Denumirea unității de curs	Responsabil de unitatea de curs	Semestrul	Numărul de ore							Evaluarea	Nr. de credite
					Total	Contact direct			AMU*	Studiul individual			
						Curs	Seminar	Laborator		asincron*	individual		
cu frecvență	S.O.10	JavaScript	Nartea N., Donu A.	2	120	30	-	30	-	-	60	E	4
cu frecvență redusă	S.O.11	JavaScript		2	120	12	-	12	-	-	96	E	4
la distanță	-	-		-	-	-	-	-	-	-	-	-	-
dual	S.O.11	JavaScript		2	120	30	-	30	-	-	60	E	4

* Activitatea de muncă la unitate – pentru învățământul dual

* Pentru învățământul cu frecvență redusă și la distanță

II. TEMATICA ŞI REPARTIZAREA ORIENTATIVĂ A ORELOR

Unități de conținut	Numărul de ore													
	Curs				Laborator				AMU	Studiul individual				
	Cu frecvență	Frecvență redusă	La distanță	Dual	Cu frecvență	Frecvență redusă	La distanță	Dual	Dual	Cu frecvență	Frecvență redusă		La distanță		Dual
											Asincron	Individual	Asincron	Individual	
Tema 1. Introducere în JavaScript. Concepte de bază **Subteme:** • Introducere în JavaScript: istoria și scopul limbajului • Primul program „Hello World” în browser • Prima aplicație simplă **Activități de laborator:** • Crearea unei pagini HTML de test și integrarea primului script JavaScript „Hello World” (inline și extern). • Exersarea afișării mesajelor în browser: utilizarea `alert()`, `console.log()` și `document.write()`. • Declararea variabilelor și constantelor folosind `let`, `const`, `var`; afișarea valorilor în consolă.	2	2		2	2	2		2		4		4			4
Tema 2. Tipuri de date, variabile și operatori. Controlul fluxului de execuție **Subteme:** • Recomandări pentru scrierea corectă a codului în JavaScript • Declarația și utilizarea variabilelor în JavaScript	2			2	2			2		4		6			4

• Tipuri de date primitive și complexe în JavaScript • Conversia și coercitivitatea tipurilor de date • Utilizarea operatorilor • Utilizarea structurilor de control pentru gestionarea fluxului de execuție • Utilizarea structurii if și ramificările condiționale • Structuri de buclă: for, while, do-while • Operatorul switch și gestionarea cazurilor multiple • Controlul fluxului cu break și continue **Activități de laborator:** • Declararea și inițializarea variabilelor de diverse tipuri (string, number, boolean, null, undefined, object, array); afișarea tipurilor de date cu typeof. • Testarea conversiei implicite și explicite a tipurilor de date (ex: `Number("123")`, `String(true)`, compararea cu `==` vs `===`). • Crearea unor expresii care utilizează operatori aritmetici, logici, de comparație și logici compuși (`&&`, `||`, `!`), și afișarea rezultatelor.															
Tema 3. Funcții **Subteme:** • Definirea și utilizarea funcțiilor în JavaScript • Declarația funcțiilor • Conceptul de „scope” și vizibilitatea variabilelor în funcții • Parametrii funcțiilor și modul de transmitere al acestora • Returnarea valorilor din funcții • Setarea valorilor implicite pentru parametrii funcțiilor **Activități de laborator:** • Definirea și apelarea funcțiilor simple: crearea unei funcții care calculează pătratul unui număr și returnează rezultatul. • Utilizarea parametrilor în funcții: scrierea unei funcții care primește numele și vârsta unei persoane și returnează un mesaj personalizat. • Exersarea conceptului de scope: definirea variabilelor în interiorul și în afara funcțiilor și testarea accesibilității acestora. • Crearea de funcții cu parametri opționali și valori implicite: exemplu cu o funcție care salută utilizatorul, cu un parametru implicit pentru limbă.	**2**	**2**		**2**	**2**	**2**		**2**		**4**		**6**			**4**
Tema 4. Funcții avansate: funcții săgeată și callback **Subteme:** • Definirea și utilizarea funcțiilor expresie • Conceptul de callback și utilizarea acestuia • Funcții săgeată (arrow functions) și diferențele față de funcțiile tradiționale • Funcții anonime și aplicabilitatea lor **Activități de laborator:** • Definirea și utilizarea funcțiilor expresie • Exersarea funcțiilor săgeată: rescrierea funcțiilor tradiționale folosind sintaxa `=>`, evidențiind diferențele de comportament față de `this`. • Implementarea unui mecanism simplu cu callback: scrierea unei funcții care primește altă funcție ca parametru și o execută în funcție de o condiție.	**2**			**2**	**2**			**2**		**4**		**6**			**4**

Tema 5. Bazele tablourilor **Subteme:** • Definirea și manipularea tablourilor în JavaScript • Destructurarea tablourilor și utilizarea operatorului de răspândire (spread) **Activități de laborator:** • Crearea și manipularea tablourilor: declararea unui tablou, accesarea elementelor, adăugarea și ștergerea elementelor cu `push()`, `pop()`, `shift()`, `unshift()`, `splice()`. • Aplicarea destructurării: extragerea valorilor din tablouri în variabile separate prin destructurare. • Utilizarea operatorului spread (`...`) pentru: concatenarea tablourilor, copierea unui tablou, extinderea parametrilor unei funcții.	**2**			**2**	**2**			**2**		**4**		**6**			**4**
Tema 6. Metodele tablourilor **Subteme:** • Metode pentru manipularea tablourilor: `map()`, `filter()`, `reduce()`, `forEach()`, … • Utilizarea ciclului `for..of` pentru parcurgerea tablourilor **Activități de laborator:** • Parcurgerea tablourilor cu `for...of`: afișarea valorilor și aplicarea unor operații simple asupra fiecărui element. • Utilizarea metodei `map()` pentru a transforma elementele unui tablou (ex: ridicarea la pătrat a fiecărui număr). • Filtrarea elementelor cu `filter()`: extragerea valorilor pare sau a elementelor ce respectă o condiție. • Crearea unui playlist muzical sub formă de tablou de obiecte JavaScript, fiecare obiect reprezentând o melodie (ex: `{ titlu, artist, gen, durata, esteFavorit }`).	**2**	**2**		**2**	**2**	**2**		**2**		**4**		**6**			**4**
Tema 7. Obiecte **Subteme:** • Ce este un obiect și utilizarea sa în JavaScript • Creează obiecte și gestionează datele complexe • Aplică destructurarea obiectelor pentru a facilita extragerea datelor • Demonstrează utilizarea obiectelor și colecțiilor de date în gestionarea informațiilor complexe • Aplica corect metodele pentru tipurile de date primitive în cadrul sarcinilor individuale • Obiectele ca tipuri de date de referință • Diferența dintre null și undefined • Exploră utilizarea `Map` pentru gestionarea structurilor de date • Utilizarea `Set` pentru gestionarea colecțiilor de valori unice • Metode pentru tipuri de date primitive (string, number, boolean) **Activități de laborator:** • Crearea unui sistem simplu de gestionare a utilizatorilor: fiecare utilizator va fi un obiect cu proprietăți precum `id`, `nume`, `email`, `rol`, `activ`. Toți utilizatorii vor fi stocați într-un tablou.	**2**	**2**		**2**	**2**	**2**		**2**		**4**		**12**			**4**

• Aplicarea destructurării obiectelor pentru a extrage rapid informații (ex: `nume` și `email`) din fiecare obiect utilizator într-un for...of. • Utilizarea unei colecții `Map` pentru asocierea ID-ului utilizatorului cu ultimele acțiuni efectuate (ex: „login”, „resetare parolă” etc.); afișarea tuturor perechilor `id → acțiune`. • Utilizarea unui `Set` pentru a stoca toate rolurile unice existente în sistem (ex: „admin”, „editor”, „utilizator”) și afișarea lor într-o listă distinctă.															
Tema 8. Obiecte avansate și clase **Subteme:** • Înțelegerea și utilizarea contextului this în obiecte și funcții • Utilizarea constructorului new pentru crearea de instanțe ale obiectelor • Utilizarea operatorului de accesare condiționată (optional chaining) pentru manipularea obiectelor • Definirea și utilizarea claselor în JavaScript **Activități de laborator:** • Definirea clasei `Inventory` care conține o proprietate `items` (tablou gol) și metode pentru: adaugă un obiect nou în inventar; parcurge și afișează fiecare item; calculează greutatea totală a tuturor obiectelor. • Testarea contextului `this`: utilizarea corectă a `this.items` în toate metodele clasei pentru a accesa și modifica inventarul. • Folosirea operatorului `?.` (optional chaining) pentru a accesa în siguranță proprietăți ale item-urilor (ex: item?.tip), evitând erori în caz de element inexistent. • Filtrarea item-urilor după tip (ex: doar „arme” sau doar „potiuni”) folosind metode de tablou (`filter()`) apelate dintr-o metodă a clasei `Inventory`.	**2**			**2**	**2**			**2**		**4**		**12**			**4**
Tema 9. Prototipuri și moștenire **Subteme:** • Conceptul de moștenire și extinderea claselor • Lanțul prototipurilor și proprietatea `__proto__` • Proprietatea prototype și comportamentul funcțiilor constructor • Moștenirea manuală folosind `Object.create()` • Moștenirea modernă cu clase ECMAScript 6 • Suprascrierea metodelor și polimorfismul **Activități de laborator:** • Extinderea sistemului de inventar: crearea constructorului Item și atașarea metodei `description()` prin Item.prototype. Testarea lanțului prototipurilor cu `__proto__`. • Creează obiecte Consumabil pe baza lui Item folosind `Object.create()` și adaugă proprietăți suplimentare (ex: `efect`). • Rescrierea structurii folosind clase ES6: clasele `Item`, `Weapon`, `Consumable`, cu moștenire și suprascrierea metodei `descriere()`. • Popularea inventarului cu instanțe din clasele derivate și testarea comportamentului polimorf prin apelul metodelor comune.	**2**			**2**	**2**			**2**		**4**		**6**			**4**

Tema 10. Module **Subteme:** • Înțelegerea și utilizarea directivei **`use strict`** • Introducerea în conceptului de module în JavaScript și rolul acestora • Exportul și importul de funcții, obiecte și variabile dintr-un modul (**`export/import`**) • Principii de organizare a codului în JavaScript folosind module **Activități de laborator:** • Refactorizează sistemul de inventar: mută clasa **`Item`** într-un fișier separat și utilizează **`export`** pentru a o face disponibilă altor fișiere. • Creează un fișier **`inventory.js`** care conține clasa **`Inventory`** și metodele sale, apoi importă-o în fișierul principal al aplicației. • Aplică **`use strict`** în toate fișierele pentru a impune reguli stricte de sintaxă și comportament. • Organizează codul în cel puțin 3 module (ex: **`item.js`**, **`inventory.js`**, **`main.js`**) și testează funcționalitatea completă prin **`import`** și **`export`**.	**2**	**2**		**2**	**2**	**2**		**2**		**4**		**4**			**4**
Tema 11. Gestionarea erorilor **Subteme:** • Tipuri de erori în JavaScript: erori runtime, erori logice • Utilizarea **`try...catch`** pentru gestionarea excepțiilor • Declanșarea excepțiilor cu **`throw`** • Tehnici de debugging și identificarea erorilor în cod • Practici bune pentru prevenirea și gestionarea erorilor **Activități de laborator:** • Adaugă validări în metodele clasei **`Inventory`** pentru a preveni introducerea de obiecte invalide. Utilizează **`throw`** pentru a declanșa erori personalizate (ex: „Greutate invalidă!”). • Înconjoară apelurile riscante cu **`try...catch`** și afișează mesaje de eroare clare pentru utilizator. • Simulează erori logice și observă diferențele față de erorile runtime (ex: index greșit, acces la proprietăți inexistente). • Folosește **`console.error()`**, **`console.trace()`** și altele pentru depanare. Exersează localizarea erorilor într-un cod modular.	**2**			**2**	**2**			**2**		**4**		**4**			**4**
Tema 12. DOM: Manipularea elementelor **Subteme:** • Introducere în mediul JavaScript și DOM-ul web • Structura și funcționarea DOM-ului (arborele DOM) • Relațiile între elementele HTML: părinte, copil, frate • Navigarea între elementele HTML în DOM (traversarea nodurilor) • Selectarea elementelor HTML cu **`getElementById`**, **`getElementsByClassName`**, și **`querySelector`** • Accesarea și manipularea proprietăților elementelor HTML **Activități de laborator:**	**2**	**2**		**2**	**2**	**2**		**2**		**4**		**6**			**4**

• Creează o pagină HTML simplă cu o structură de bază pentru afișarea inventarului (ex: o listă `<ul>` goală și un buton „Afișează inventarul”). • Selectează elementele din pagină folosind `getElementById`, `getElementsByClassName` și `querySelector`, apoi modifică conținutul lor din JavaScript. • Afișează lista de obiecte din inventar în DOM, parcurgând tabloul de itemi și generând elemente `<li>` pentru fiecare. • Exersează navigarea între elementele HTML (părinte, copil, …) folosind `parentNode`, `children`, `nextElementSibling` etc.															
Tema 13. DOM: Gestionarea evenimentelor **Subteme:** • Introducere în evenimentele JavaScript și ascultătorii de evenimente • Propagarea evenimentelor: captare și bubbling • Delegarea evenimentelor pentru optimizarea gestionării evenimentelor • Anularea acțiunilor implicite ale browserului (`preventDefault()`) **Activități de laborator:** • Adaugă un formular HTML pentru introducerea unui nou item (nume, tip, greutate) și atașează un ascultător de eveniment pe butonul de adăugare. • Folosește `preventDefault()` pentru a preveni reîncărcarea paginii la trimiterea formularului și adaugă itemul în lista de inventar. • Exersează propagarea evenimentelor: adaugă un eveniment pe un container și observă comportamentul bubbling/capturing. • Aplică delegarea evenimentelor pe lista de itemi: detectează click pe un `<li>` și afișează detalii sau elimină itemul selectat.	**2**		**2**	**2**			**2**		**4**		**6**			**4**
Tema 14. Introducere în programarea asincronă în JS **Subteme:** • Ce este programarea asincronă? Diferența dintre programarea sincronă și asincronă • Utilizarea funcțiilor de tip callback pentru gestionarea sarcinilor asincrone • Introducerea conceptului de `Promise` în JavaScript și cum funcționează • Utilizarea metodelor `then()` și `catch()` pentru a gestiona succesul și erorile în `Promise` • Introducerea `async` și `await` pentru simplificarea gestionării promisiunilor • Cum `async` și `await` fac codul asincron mai clar și ușor de întreținut • Gestionarea erorilor în operațiuni asincrone folosind `try...catch` **Activități de laborator:** • Exersează apeluri asincrone simple folosind `setTimeout()` și funcții callback pentru a înțelege ordinea execuției. • Creează o funcție care returnează un Promise și folosește `then()` și `catch()` pentru a trata succesul și erorile.	**4**		**4**	**4**			**4**		**8**		**12**			**8**

• Refactorizează aceeași logică folosind **`async / await`** și gestionează erorile cu **`try...catch`**. • Folosește **`fetch()`** pentru a obține date dintr-un API public (ex: JSONPlaceholder), afișează titlurile într-o listă HTML.															
Total	**30**	**12**	**0**	**30**	**30**	**12**	**0**	**30**	**0**	**60**	**0**	**96**	**0**	**0**	**60**

III. COMPETENŢE GENERALE, PROFESIONALE ŞI REZULTATELE ÎNVĂȚĂRII

Competențe generale și profesionale	**Rezultate ale învățării conform nivelului CNC** *Absolventul/candidatul la atribuirea calificării poate:*
CG 2. Operarea cu concepte de bază din ştiinţa calculatoarelor, tehnologia informaţiei şi comunicaţiilor	**RÎ3.** aplica conceptele din știința calculatoarelor, tehnologia informației și comunicațiilor pentru proiectarea și administrarea sistemelor informaționale. **RÎ4.** dezvolta sisteme informaționale folosind cunoștințe referitoare la limbaje, medii și tehnologii de programare și instrumente de proiectare.
CP 1. Specificarea funcționării sistemului informatic utilizând metode, modele și algoritmi	**RÎ10.** elabora modelul constructiv-funcțional prin utilizarea tehnicilor, modelelor și algoritmilor din domeniu, aplicând metode de modelare, sinteză, analiză, precum și optimizare a sistemelor.
CP 2. Proiectarea sistemelor și integrarea componentelor unui sistem informatic	**RÎ11.** proiecta aplicații sotware de uz general și dedicat, prin aplicarea metodelor de modelare, sinteză, analiză și optimizare a sistemelor informatice. **RÎ12.** dezvolta conceptul de realizare și funcționare a sistemului proiectat, precum și a unor ansambluri parțiale integrate în subsistemul informatic.
CP 3. Validarea aplicațiilor informatice	**RÎ14.** realiza simulări pentru validarea corectitudinii modelelor și algoritmilor, asigurând compatibilitatea componentelor cu cerințele și obiectivele stabilite.
CP 4. Mentenanța și optimizarea sistemului informatic	**RÎ15.** planifica lucrările de mentenatță pentru a asigura buna funcționare a sistemului informatic.

IV. UNITĂŢI DE ÎNVĂȚARE

Tema 1. Introducere în JavaScript. Concepte de bază Rezultatele învățării preconizate a fi atinse: **RÎ3, RÎ4, RÎ11**		
Cunoștințe / unități de conținut	**Abilități**	**Responsabilitate și autonomie**
Termeni-cheie: JavaScript, Browser, Istoria JavaScript, DOM, „Hello World”, Variabile,	• Identifică și explică rolul limbajului JavaScript în	• Demonstrează autonomie în realizarea primelor exerciții JavaScript prin

Console **Unități de conținut:** • Introducere în JavaScript: istoria și scopul limbajului • Primul program „Hello World” în browser • Prima aplicație simplă	dezvoltarea aplicațiilor web. • Redactează și execută scripturi JavaScript simple într-un mediu de tip browser. • Utilizează consola browserului pentru testare și depanare de bază. • Creează o aplicație minimă demonstrativă folosind DOM și interacțiuni simple. • Aplică convenții corecte de denumire și inițializare a variabilelor.	implementarea soluțiilor cu resurse limitate. • Evaluează soluțiile proprii și pe cele ale colegilor pentru a îmbunătăți procesele de învățare.
Tema 2. Tipuri de date, variabile și operatori. Controlul fluxului de execuție Rezultatele învățării preconizate a fi atinse: **RÎ4, RÎ10, RÎ11, RÎ14**		
Cunoștințe / unități de conținut	**Abilități**	**Responsabilitate și autonomie**
Termeni-cheie: tipuri de date, variabile, const, let, var, operatori aritmetici, operatori logici, structuri de control, for, if, switch, while, dowhile, break, continue **Unități de conținut:** • Recomandări pentru scrierea corectă a codului în JavaScript • Declarația și utilizarea variabilelor în JavaScript • Tipuri de date primitive și complexe în JavaScript	• Identifică și declară variabilele folosind const, let și var în funcție de context. • Utilizează operatori logici și aritmetici pentru a exprima condiții și calcule. • Proiectează și implementează structuri condiționale și repetitive pentru controlul logicii programului. • Aplică transformări de tip și gestionează coerent datele în expresii mixte.	• Demonstrează abilitatea de a aplica corect structurile de control în funcție de necesitățile aplicației. • Evaluează soluțiile propuse pentru a optimiza execuția programului prin utilizarea corectă a buclelor și a condițiilor.

• Conversia și coercitivitatea tipurilor de date • Utilizarea operatorilor • Utilizarea structurilor de control pentru gestionarea fluxului de execuție • Utilizarea structurii if și ramificările condiționale • Structuri de buclă: for, while, do-while • Operatorul switch și gestionarea cazurilor multiple • Controlul fluxului cu break și continue • Gestionarea problemelor legate de fluxul execuției programului identificarea erorilor logice și corectarea acestora prin intermediul structurii de control	• Simulează scenarii și identifică erori logice sau de structură în fluxul aplicației.	
Tema 3. Funcții Rezultatele învățării preconizate a fi atinse: **RÎ4, RÎ10, RÎ11, RÎ12**		
Cunoștințe / unități de conținut	**Abilități**	**Responsabilitate și autonomie**
Termeni-cheie: funcții, parametri, valoare de retur, scope, variabile locale și globale, valori implicite **Unități de conținut:** • Definirea și utilizarea funcțiilor în JavaScript • Declarația funcțiilor	• Creează și apelează funcții pentru a rezolva sarcini specifice. • Explică și aplică conceptul de „scope” pentru gestionarea corectă a vizibilității variabilelor. • Utilizează parametri pentru a transmite date în funcții și	• Demonstrează abilitatea de a structura codul utilizând funcții modulare și reutilizabile. • Evaluează și îmbunătățește eficiența codului prin utilizarea corectă a parametrilor și a valorilor returnate din funcții.

• Conceptul de „scope” și vizibilitatea variabilelor în funcții • Parametrii funcțiilor și modul de transmitere al acestora • Returnarea valorilor din funcții • Setarea valorilor implicite pentru parametrii funcțiilor	pentru a returna rezultate din acestea • Optimizează funcțiile prin utilizarea valorilor implicite pentru parametrii atunci când este necesar.	
Tema 4. Funcții avansate: funcții săgeată și callback Rezultatele învățării preconizate a fi atinse: **RÎ10, RÎ11, RÎ12, RÎ14**		
Cunoștințe / unități de conținut	**Abilități**	**Responsabilitate și autonomie**
Termeni-cheie: funcții expresie, funcții săgeată, callback, funcții anonime **Unități de conținut:** • Definirea și utilizarea funcțiilor expresie • Conceptul de callback și utilizarea acestuia • Funcții săgeată (arrow functions) și diferențele față de funcțiile tradiționale • Funcții anonime și aplicabilitatea lor	• Creează funcții de tip callback pentru a implementa logica asincronă • Utilizează funcții săgeată pentru a simplifica scrierea funcțiilor în contexte moderne • Analizează diferențele dintre funcțiile săgeată și cele tradiționale pentru a aplica cea mai potrivită soluție • Explică utilizarea funcțiilor anonime și când este benefică utilizarea acestora	• Demonstrează abilitatea de a folosi funcții avansate pentru structuri modulare și eficiente de cod • Optimizează fluxul de lucru asincron prin funcții callback și funcții săgeată
Tema 5. Bazele tablourilor Rezultatele învățării preconizate a fi atinse: **RÎ4, RÎ10, RÎ11**		
Cunoștințe / unități de conținut	**Abilități**	**Responsabilitate și autonomie**

Termeni-cheie: tablouri(arrays), desctructurare, operatorul de răspândire (spread operator) **Unități de conținut:** • Definirea și manipularea tablourilor în JavaScript • Destructurarea tablourilor și utilizarea operatorului de răspândire (spread)	• Creează și manipulează tablouri pentru stocarea și prelucrarea datelor. • Aplică destructurarea pentru extragerea rapidă a valorilor din tablou. • Utilizează operatorul de răspândire pentru operații avansate asupra tablourilor. • Combină conceptele de funcții, bucle și tablouri pentru a implementa logică aplicativă practică. • Identifică structura optimă de tablou pentru diverse tipuri de date și scopuri.	• Demonstrează autonomie în utilizarea tehnicilor moderne de manipulare a datelor în tablouri. • Respectă principiile de eficiență și claritate în scrierea codului care implică arrays. • Manifestă responsabilitate în alegerea metodelor de procesare a datelor în funcție de contextul aplicației.
Tema 6. Metodele tablourilor Rezultatele învățării preconizate a fi atinse: **RÎ4, RÎ10, RÎ11, RÎ12**		
Cunoștințe / unități de conținut	**Abilități**	**Responsabilitate și autonomie**
Termeni-cheie: metodele tablourilor, bucla for..of **Unități de conținut:** • Metode pentru manipularea tablourilor: `map()`, `filter()`, `reduce()`, `forEach()`, … • Utilizarea ciclului `for..of` pentru parcurgerea tablourilor	• Enumeră și explică metodele de procesare a tablourilor (`map`, `filter`, `reduce`, `forEach`) și bucla `for...of`. • Aplică metode funcționale pentru a transforma, filtra și agrega datele din tablouri. • Utilizează `for...of` și forEach pentru a parcurge și analiza elementele unui tablou. • Compară metodele de iterare din punct de vedere al eficienței, clarității și	• Manifestă autonomie în alegerea soluțiilor moderne de prelucrare a tablourilor. • Demonstrează responsabilitate în utilizarea corectă și eficientă a metodelor funcționale. • Respectă bunele practici în structurarea codului care implică procesarea datelor secvențiale.

	aplicabilității în diverse scenarii. • Combină mai multe metode de procesare într-un flux coerent pentru a rezolva probleme practice. • Evaluează calitatea și performanța soluțiilor implementate, justificând alegerea tehnicii folosite.	
Tema 7. Obiecte Rezultatele învățării preconizate a fi atinse: **RÎ4, RÎ10, RÎ11, RÎ14**		
Cunoștințe / unități de conținut	**Abilități**	**Responsabilitate și autonomie**
Termeni-cheie: obiecte, referință, destructurare, null, undefined, Map, Set **Unități de conținut:** • Ce este un obiect și utilizarea sa în JavaScript • Obiectele ca tipuri de date de referință • Destructurarea obiectelor • Diferența dintre `null` și `undefined` • Utilizarea `Map` • Utilizarea `Set` pentru gestionarea colecțiilor de valori unice • Metode pentru tipuri de date primitive (`string`, `number`, `boolean`)	• Explică structura și comportamentul obiectelor ca tipuri de date de referință. • Creează și modifică obiecte utilizând proprietăți, metode și destructurare. • Compară și gestionează valorile null și undefined în cadrul fluxului logic al programului. • Utilizează metodele specifice pentru tipurile primitive în construirea funcționalităților aplicative. • Evaluează și selectează structura de date potrivită (obiect, Map, Set) în funcție de scopul aplicație	• Demonstrează utilizarea obiectelor și colecțiilor de date în gestionarea informațiilor complexe • Aplica corect metodele pentru tipurile de date primitive în cadrul sarcinilor individuale

Tema 8. Obiecte avansate și clase Rezultatele învățării preconizate a fi atinse: **RÎ4, RÎ10, RÎ11, RÎ12, RÎ14**		
Cunoștințe / unități de conținut	**Abilități**	**Responsabilitate și autonomie**
Termeni-cheie: obiecte avansate, `this`, `constructor`, `new`, optional chaining, clase **Unități de conținut:** • Înțelegerea și utilizarea contextului this în obiecte și funcții • Utilizarea constructorului new pentru crearea de instanțe ale obiectelor • Utilizarea operatorului de accesare condiționată (optional chaining) pentru manipularea obiectelor • Definirea și utilizarea claselor în JavaScript	• Explică funcționarea contextului `this` în diferite tipuri de funcții și metode. • Creează instanțe de obiecte utilizând constructori și operatorul `new`. • Utilizează operatorul `?.` pentru a evita erorile la accesarea proprietăților inexistente. • Proiectează și implementează clase pentru modelarea entităților aplicației. • Combină clase, metode și instanțe pentru organizarea logicii aplicației în stil orientat pe obiect. • Evaluează avantajele programării orientate pe obiect față de structuri funcționale sau imperative.	• Demonstrează abilitatea de a structura aplicațiile folosind clase și obiecte avansate fără asistență direct • Îmbunătățește soluțiile proprii folosind accesarea condiționată și gestionarea eficientă a instanțelor de obiecte
Tema 9. Prototipuri și moștenire Rezultatele învățării preconizate a fi atinse: **RÎ4, RÎ10, RÎ11, RÎ12**		
Cunoștințe / unități de conținut	**Abilități**	**Responsabilitate și autonomie**
Termeni-cheie: prototipuri, moștenire, `__proto__`, prototype, `Object.create()`, clase ECMAScript, suprascriere, polimorfism	• Explică mecanismul de moștenire prototipală și rolul lanțului `__proto__`. • Creează structuri de obiecte care folosesc moștenire	• Demonstrează autonomie în alegerea abordării adecvate pentru reutilizarea codului: prototipală sau bazată pe clase.

Unități de conținut: • Conceptul de moștenire și extinderea claselor • Lanțul prototipurilor și proprietatea __proto__ • Proprietatea prototype și comportamentul funcțiilor constructor • Moștenirea manuală folosind Object.create() • Moștenirea modernă cu clase ECMAScript 6 • Suprascrierea metodelor și polimorfismul	manuală și prototipuri personalizate. • Utilizează moștenirea cu clase moderne pentru extinderea comportamentului. • Suprascrie metode în clase derivate și aplică principiile polimorfismului în cod. • Compară avantajele și limitările diferitelor abordări de moștenire în JavaScript. • Proiectează ierarhii simple de clase pentru organizarea logicii aplicației.	• Manifestă interes în explorarea relațiilor între obiecte și aplicarea principiilor de moștenire în structurarea codului.
Tema 10. Module Rezultatele învățării preconizate a fi atinse: **RÎ4, RÎ11, RÎ12, RÎ14**		
Cunoștințe / unități de conținut	**Abilități**	**Responsabilitate și autonomie**
Termeni-cheie: `use strict`, module, `export`, `import`, organizarea codului **Unități de conținut:** • Înțelegerea și utilizarea directivei use strict pentru o mai bună gestionare a erorilor • Introducerea în conceptului de module în JavaScript și rolul acestora • Exportul și importul de funcții, obiecte și variabile dintr-un modul (export/import)	• Explică rolul directivei use strict și efectele sale asupra interpretării codului. • Aplică conceptele de modularizare în organizarea codului. • Utilizează corect instrucțiunile export și import pentru a împărți codul în fișiere reutilizabile. • Creează module clare, bine delimitate, pentru separarea logicii funcționale a aplicației.	• Demonstrează abilitatea de a organiza codul în mod autonom folosind module și respectând principiile de modularitate • Aplică în mod independent bune practici de organizare a codului și reutilizarea acestuia prin structuri modulare

• Principii de organizare a codului în JavaScript folosind module	• Analizează structura proiectului și reorganizează codul pentru a respecta principiile de scalabilitate și mentenanță. • Evaluează avantajele modularizării în contextul lucrului colaborativ și al reutilizării codului.	
Tema 11. Gestionarea erorilor Rezultatele învățării preconizate a fi atinse: **RÎ4, RÎ10, RÎ14, RÎ15**		
Cunoștințe / unități de conținut	**Abilități**	**Responsabilitate și autonomie**
Termeni-cheie: Gestionarea erorilor, `try...catch`, `throw`, debugging, erori runtime, erori logice, manipularea excepțiilor **Unități de conținut**: • Tipuri de erori în JavaScript: erori runtime, erori logice • Utilizarea try...catch pentru gestionarea excepțiilor • Declanșarea excepțiilor cu throw • Tehnici de debugging și identificarea erorilor în cod • Practici bune pentru prevenirea și gestionarea erorilor	• Identifică și clasifică erorile întâlnite în aplicații (erori runtime și logice) • Aplică blocurile `try...catch` pentru a prinde și gestiona excepțiile în mod eficient • Utilizează instrucțiunea throw pentru a declanșa erori personalizate și a gestiona fluxul aplicației • Depanează codul folosind tehnici de debugging pentru a identifica sursele erorilor • Propune soluții eficiente pentru a preveni erorile comune în aplicațiile JavaScript • Implementează sisteme de logging pentru a monitoriza erorile în aplicațiile de producție	• Demonstrează abilitatea de a identifica și gestiona erorile în codul aplicației fără supervizare • Asigură continuitatea fluxului de execuție printr-o gestionare corectă a excepțiilor • Aplică în mod responsabil metodele de prevenire a erorilor și asigură calitatea codului

<table>
<tr><th colspan="3">Tema 12. DOM: Manipularea elementelor
Rezultatele învățării preconizate a fi atinse:</th></tr>
<tr><th>Cunoștințe / unități de conținut</th><th>Abilități</th><th>Responsabilitate și autonomie</th></tr>
<tr>
<td>Termeni-cheie: DOM (Document Object Model), noduri, <code>getElementById</code>, <code>querySelector</code>, traversare DOM, relații între elemente HTML, proprietăți ale elementelor

Unități de conținut:
Introducere în mediul JavaScript și DOM-ul web
Structura și funcționarea DOM-ului (arborele DOM)
Relațiile între elementele HTML: părinte, copil.
Navigarea între elementele HTML în DOM (traversarea nodurilor)
Selectarea elementelor HTML cu <code>getElementById</code>, <code>getElementsByClassName</code>, și <code>querySelector</code>
Accesarea și manipularea proprietăților elementelor HTML</td>
<td>Descrie structura DOM-ului și identifică poziția unui element în arborele HTML.
Utilizează metode de selecție pentru a accesa elemente specifice din document.
Traversează DOM-ul pentru a interacționa cu noduri înrudite (părinte, copil, frate).
Modifică atribute, proprietăți și conținut al elementelor HTML folosind JavaScript.
Aplică conceptele în crearea de interfețe interactive și dinamice.
Evaluează eficiența metodelor de selecție și manipulare în funcție de contextul aplicației.</td>
<td>Demonstrează abilitatea de a lucra independent cu DOMul pentru a modifica structura și stilul elementelor HTML
Aplică metodele de traversare și selectare a elementelor DOM pentru a crea interacțiuni dinamice fără asistență
Îmbunătățește performanța aplicației prin manipularea eficientă a elementelor DOM</td>
</tr>
<tr><th colspan="3">Tema 13. DOM: Gestionarea evenimentelor
Rezultatele învățării preconizate a fi atinse: RÎ4, RÎ10, RÎ11, RÎ12</th></tr>
<tr><th>Cunoștințe / unități de conținut</th><th>Abilități</th><th>Responsabilitate și autonomie</th></tr>
<tr>
<td>Termeni-cheie: Evenimente, ascultători de evenimente, propagare, captare, bubbling, delegarea evenimentelor, acțiuni</td>
<td>Explică mecanismul de propagare a evenimentelor și diferențiază captarea de bubbling.</td>
<td>Demonstrează abilitatea de a gestiona evenimentele DOM în mod eficient și autonomy</td>
</tr>
</table>

implicite ale browserului **Unități de conținut**: • Introducere în evenimentele JavaScript și ascultătorii de evenimente • Propagarea evenimentelor: captare și bubbling • Delegarea evenimentelor pentru optimizarea gestionării evenimentelor • Anularea acțiunilor implicite ale browserului (`preventDefault()`)	• Creează ascultători de evenimente pentru elemente HTML și definește funcții de tratament. • Aplică delegarea evenimentelor pentru eficiența gestionării în structuri dinamice. • Anulează comportamentul implicit al browserului în contexte relevante (formulare, linkuri). • Integrează tratarea evenimentelor în logica aplicației pentru a crea interacțiuni personalizate. • Evaluează impactul și performanța diferitelor tehnici de gestionare a evenimentelor.	• Aplica corect metodele de propagare și delegare a evenimentelor pentru a optimiza interacțiunile cu utilizatorul
Tema 14. Introducere în programarea asincronă în JS Rezultatele învățării preconizate a fi atinse: **RÎ4, RÎ10, RÎ11, RÎ12, RÎ14**		
Cunoștințe / unități de conținut	**Abilități**	**Responsabilitate și autonomie**
Termeni-cheie: Programare asincronă, callback, Promise, `then()`, `catch()`, `async`, `await` **Unități de conținut**: • Ce este programarea asincronă? Diferența dintre programarea sincronă și asincronă	• Explică conceptele de bază ale programării asincrone și identifică momentele în care este necesară. • Implementează funcții asincrone folosind callback, Promise și `async/await`. • Utilizează `then()` și `catch()` pentru tratarea rezultatelor și erorilor din promisiuni.	• Demonstrează abilitatea de a implementa sarcini asincrone folosind promisiuni și funcții callback • Aplică independent conceptele de programare asincronă pentru a gestiona fluxurile de date și operațiunile în aplicații web

• Utilizarea funcțiilor de tip callback pentru gestionarea sarcinilor asincrone • Introducerea conceptului de Promise în JavaScript și cum funcționează • Utilizarea metodelor `then()` și `catch()` pentru a gestiona succesul și erorile în Promise • Introducerea async și await pentru simplificarea gestionării promisiunilor • Cum async și await fac codul asincron mai clar și ușor de întreținut • Gestionarea erorilor în operațiuni asincrone folosind `try...catch`	• Aplică `async` și `await` pentru scrierea unui cod asincron clar și predictibil. • Gestionează corect excepțiile apărute în operațiuni asincrone prin blocuri `try...catch`. • Evaluează avantajele fiecărei abordări în funcție de complexitatea sarcinii asincrone.	

V. STUDIUL INDIVIDUAL AL STUDENTULUI

Fiecare student va dezvolta o aplicație web interactivă utilizând JavaScript, care va demonstra utilizarea conceptelor învățate pe parcursul cursului.

Nr	Produsul preconizat	Strategii de realizare	Termen de realizare
1	Aplicație web interactivă	**1.** Identificarea unei teme relevante **2.** Proiectarea structurii aplicației **3.** Dezvoltarea funcționalităților de bază **4.** Testarea și documentarea proiectului	Nu mai târziu de o săptămână înainte de încheierea semestrului

Criterii de evaluare a produsului

Criterii de evaluare	0 puncte	1 puncte	2 puncte	3 puncte	4 puncte
Formularea temei și selectarea unei structuri corespunzătoare a aplicației	Nu este formulată tema și nu este selectată o structură corespunzătoare pentru aplicație	Tema este formulată întrun mod neadecvat sau nu este selectată o structură adecvată pentru aplicație	Tema este parțial formulată și este selectată o structură adecvată cu abateri semnificative	Tema este formulată întrun mod adecvat, iar structura selectată este corespunzătoare, dar una dintre ele nu respectă strategiile de realizare	Tema este formulată adecvat, structura selectată este potrivită și respectă toate cerințele strategiilor de realizare
Conformitatea cu standardele de codare	Codul nu respectă standardele și nu este structurat corespunzător	Codul parțial respectă standardele, dar lipsesc structurarea corespunzătoare	Codul parțial respectă standardele, dar există abateri semnificative	Codul respectă standardele, dar unul dintre acestea nu este respectat	Codul respectă standardele și respectă toate cerințele acestora
Claritatea și comentarea codului	Codul este greu de citit și nu conține comentarii	Codul este slab citit și conține un număr limitat de comentarii	Codul este ușor citit, dar există dificultăți și/sau comentariile sunt limitate	Codul este ușor citit, dar există unele dificultăți și/sau comentariile sunt incomplete.	Codul este ușor citit, conține comentarii detaliate și explicații clare ale logicii și soluțiilor în cod.
Realizarea cerințelor stabilite pentru	Nu sunt îndeplinite cerințele sau sunt încălcate	Cerințele sunt îndeplinite în mod limitat sau cu	Cerințele sunt îndeplinite parțial, dar	Cerințele sunt îndeplinite, dar cu unele	Cerințele sunt în mare măsură îndeplinite,

proiectul individual	în mod semnificativ	încălcări evidente	cu abateri semnificative	nerespectări sau erori notabile	respectând obiectivele și standardele cerute
Respectarea structurii raportului	Raportul nu este structurat sau lipsesc componentele / plagiat	Lipsesc mai multe compartimente sau componente	Una din părțile componente lipsește sau este parțial plagiată	Respectă toate părțile componente ale raportului, dar una din ele nu este realizată conform strategiilor de realizare	Respectă toate părțile componente ale raportului
Completitudinea și corectitudinea redactării surselor bibliografice și citatelor	Lipsesc sursele bibliografice sau acestea sunt scrise incorect	Completarea incompletă sau incorectă a mai multor surse bibliografice și citate	Completare parțială sau parțial corectă a surselor bibliografice și citatelor	Sursele bibliografice și citatele sunt indicate corect, dar există abateri	Completarea și corectitudinea a bibliografiei și citatelor
Prezentarea produsului în termenele stabilite	Produsul nu a fost prezentat	Produsul a fost prezentat cu o întârziere de 7 sau mai multe zile	Produsul a fost prezentat cu o întârziere de 3-6 zile	Produsul a fost prezentat cu o întârziere de 1-2 zile	Produsul a fost prezentat în termenii stabiliți și în mod complet

VI. SUGESTII METODOLOGICE DE PREDARE-ÎNVĂȚARE-EVALUARE

Activitatea de predare-învățare la disciplina „JavaScript” este organizată în cadrul orelor de curs, laborator și lucru individual (activitate independentă a studentului).

Pe parcursul cursului, se vor propune lucrări de laborator și teme pentru acasă, în conformitate cu unitățile de conținut planificate.

La cursuri sunt utilizate metode tradiționale și unele specifice de predare: prezentări, rezolvarea comună a anumitor probleme, interacțiune interactivă cu studenții și rezolvarea sarcinilor propuse.

În cadrul orelor de laborator și în lucrările independente sunt propuse sarcini ce țin de întreaga materie studiată, având scopul de a valorifica și aprofunda cunoștințele, de a stimula activitatea de învățare, centrată pe student. tudenții rezolvă aceste exerciții pe calculatoarele din sala de laborator sau în format scris, iar apoi explică metodele și modalitățile de rezolvare. Aceste exerciții au ca scop înțelegerea mai profundă a materialului și dezvoltarea abilităților practice în domeniul dezvoltării web.

Timpul alocat pentru lucrul independent al studenților este destinat pregătirii pentru orele de curs și laborator, rezolvării sarcinilor curente și realizării lucrărilor independente, cum ar fi studii de caz sau proiecte de cercetare.

Evaluarea în cadrul disciplinei „JavaScript" se va realiza prin metode formative și sumative:

Evaluarea formativă: include activitatea la orele de laborator, teste orale sau scrise curente, și lucrările practice realizate pe parcursul semestrului.

Evaluarea sumativă: constă în prezentarea și evaluarea proiectului individual la finalul semestrului,care reflectă capacitatea studentului de a aplica cunoștințele în dezvoltarea unei aplicații funcționale.

Nota general (NG) la disciplină însumează nota de la examen (NE) și cea semestrială (NS), în proporție de 40 și, respectiv, 60 la sută ($\boldsymbol{NG = 0.4 * NE + 0.6 * NS}$), și apreciază gradul de corespundere cu finalitățile scontate: cunoștințele și competențele acumulate, abilitatea de a aplica cunoștințele, gradul de integrare a cunoștințelor de către studenți etc.

La studii cu frecvență redusă, *nota general (NG)* la disciplină însumează nota de la examen (NE) și cea semestrială (NS), în proporție de 50 și, respectiv, 50 la sută ($\boldsymbol{NG = 0.5 * NE + 0.5 * NS}$).

Responsabilitatea principală pentru evaluarea teoretică revine titularului de curs, care organizează testările și examenul din sesiune. Asistentul de laborator este responsabil pentru evaluarea activităților practice și a lucrului individual, monitorizând progresul studenților și introducând notele în registrul electronic al grupei academice.

În cazul susținerii examenului în sesiunea suplimentară, responsabilitățile privind evaluarea lucrărilor se stabilesc prin acord între titularul de curs și asistentul de laborator.

BIBLIOGRAFIE RECOMANDATĂ

1. D. Flanagan. JavaScript The Definitive Guide. Seven edition, 2020
2. M. Haverbeke. Eloquent JavaScript. Third edition, 2018.
3. The Modern JavaScript Tutorial. Javascript Info (Sursă electronică). URL: https://javascript.info/
4. JavaScript — Dynamic client-side scripting. mdn web docs (Sursă electronică). URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript