**Tehnici Web**

Esti logat drept justicebringer

[Iesi din cont](http://irinaciocan.ro/tehnici_web/logout.php)

* [Principal](http://irinaciocan.ro/tehnici_web/index.php)

* [Profil](http://irinaciocan.ro/tehnici_web/inregistrare.php)

* [Regulament](http://irinaciocan.ro/tehnici_web/regulament.php)

* Proiect HTML+CSS

* Laboratoare

* M.I.

* [Note](http://irinaciocan.ro/tehnici_web/note.php)

* Proiect JS

* Recapitulare

* Anexe

**Proiect JavaScript**

**Baremul este acum actualizat - am trecut si cerintele de node**

**Important! Despre punctaj. Pentru majoritatea obiectivelor vedeti un punctaj minim si unul maxim. Punctajul minim se refera la introducerea in forma cea mai simpla a acelui element in proiect. Punctajul creste spre valoarea maxima cu cat obiectivul respectiv este reprezentat intr-un mod mai variat si mai complex. Uneori pentru efecte realmente deosebite (din punct de vedere al creativitatii sau complexitatii) se poate da chiar mai mult decat e trecut ca punctaj maxim (sub forma de bonus).**

Acest proiect nu este de sine statator ci este o continuare a proiectului de html si css. Deci functionalitatile implementate trebuie sa fie in concordanta cu tema primului proiect.

Descrierea proiectului si modul de realizare a cerintelor minimale (unde vor fi incluse) trebuie specificate completand formularul de descriere a proiectului de JavaScript. Imi rezerv dreptul de a completa si modifica formularul in cazul in care studentul scrie in mod neclar ce doreste sa realizeze sau sunt multi studenti care au cerinte similare, sau aplicatia descrisa de student seamana prea mult cu aplicatii existente deja pe internet sau implementate in proiectele din anii anteriori. Se va anunta la laborator si pe grupul de Facebook cand consideram ca temele au un enunt final.

Lista cu cerintele eliminatorii o gasiti pe [site-ul profesorului titular de curs](https://sites.google.com/site/fmitehniciweb/home#TOC-Detalii-despre-proiect). Acestea apar si in barem, uneori impartite pe mai multe cerinte mici.

**Detalii de implementare pentru aplicatie**

**Impartirea proiectului pe foldere**

Fiecare tip de resursa ar trebui sa fie in folderul sau. Imaginile, de exemplu, in folderul "imagini", fisierele audio in folderul "audio", idem pentru fisierele video. De asemenea, fisierele de configurare in format xml ar putea fi puse intr-un folder numit "configurare" sau "xml". Daca exista si css-uri specifice jocului, acestea vor fi, de exemplu, in folderul css.

**Fisiere de configurare xml si obiectul Loader**

Un joc in general are nevoie de anumite resurse precum: imagini, fisiere video, fisiere audio etc. Aceste resurse trebuie incarcate inainte de incepe jocul efectiv, pentru ca altfel, cand e nevoie de ele s-ar putea sa nu fi fost inca downloadate de pe server. Un mod eficient de trata problema e sa aveti un obiect special, numit Loader, care are o metoda Loader.parseResources(fisierxml) care se ocupa de incarcarea resurselor din xml si afisarea unui loading screen. Loading screen-ul ar consta dintr-un element *progress* care creste in valoare la fiecare resursa incarcata. Mai jos aveti un exemplu de fisier xml:

**Fisier xml de resurse**

selecteaza textul<?xml version="1.0" encoding="utf-8" ?>  
<resurse>  
    <imagini>  
        <imagine tip="adversar" w="10" h="10" id="fant\_mica" aparitie="multipla">  
            img/adversari/fantoma\_mica.png  
        </imagine>  
        <imagine tip="adversar" w="20" h="40" id="fant\_mare" aparitie="multipla">  
            img/adversari/fantoma\_mare.png  
        </imagine>  
        <imagine tip="adversar" w="20" h="20" id="fant\_rea" aparitie="multipla">  
            img/adversari/fantoma\_rea.png  
        </imagine>  
        <imagine tip="jucator" w="20" h="20" id="jucator" aparitie="singulara">  
            img/jucator/erou.png  
        </imagine>  
    </imagini>  
    <sunete>  
        <sunet id="strigat">  
            audio/buhuhuhuahahahahaha.ogg  
        </sunet>  
        <sunet id="valeu">  
            audio/jucator/valeu.ogg  
        </sunet>  
    </sunete>  
    <fisiere\_video>  
        <video id="cutscene">  
            video/bataie\_fantome.ogv  
        </video>  
    </fisiere\_video>  
</resurse>

Pentru acest fisier, de exemplu, vom crea o propietate a Loader-ului numita *imagini*. Pentru fiecare set de informatii pentru cate o imagine vom crea un element DOM de tip imagine si vom adauga in obiectul imagini o proprietate cu id-ul resursei respective si avand ca valoare obiectul DOM asociat. La fel vom face si pentru audio si video.

Pentru fiecare tip de resursa vom avea si cate o functie de tip *get* care primeste id-ul resursei:

* Loader.getImage(id\_xml) - va intoarce elementul DOM asociat si penttru imaginile cu atributul aparitie="multipla" va crea un element nou de tip imagine (in asteptare). Facem acest lucru deoarece dorim sa avem aceeasi imagine de mai multe ori in pagina. Daca aparitie="singulara" inseamna ca stim ca imaginea o sa apara o singura data asa ca nu e nevoie sa creem mai multe elemente DOM de acelasi fel.
* Loader.getAudio(id\_xml) - va intoarce elementul audio asociat (fara a mai crea altul)
* Loader.getVideo(id\_xml) - va intoarce elementul video asociat (fara a mai crea altul, decat daca vreti sa rulati in paralel acelasi videoclip de mai multe ori)

In urma executiei functiei parseResources() obiectul Loader ar trebui sa aiba pentru fiecare tip de date cate un obiect in care proprietatile sunt egale cu id-urile elementelor iar valorile egale cu obiectele asociate.

De asemenea dorim sa avem stocate diverse date referitoare la joc; date pe care nu am dori sa le hardcodam (sa le scriem valorile efective in cod). Pentru acest lucru veti folosi fisiere xml.

Un exemplu de fisier de date e fisierul de mesaje. Putem avea mai multe cazuri in care vrem sa dam acelasi mesaj (de exemplu, mesajul care anunta jucatorul caa a pierdut jocul e dat si cand a ajuns cu sanatatea la zero dar si cand a atins marginea chenarului jocului). Am vrea sa nu il hardcodam"" pentru ca poate mai tarziu dorim sa il schimbam si ar trebui sa modificam fiecare aparitie a sa in cod. De asemenea, daca vrem sa putem seta jocul pentru diverse limbi, am vrea o metoda rapida si eficienta de a obtine mesajul potrivit pentru fiecare situatie (deci nu sa facem un set de if-uri care verifica limba setata, in cazul fiecarui mesaj de afisat)

**Fisier xml de mesaje**

selecteaza textul<?xml version="1.0" encoding="utf-8" ?>  
<lang>  
    <mesaje lang="ro">  
        <mesaj id="fail">  
            Ai dat-o-n bara.  
        </mesaj>  
        <mesaj id="castig">  
            Yaaaaay ai castigat!  
        </mesaj>  
        <mesaj id="pierdut">  
            Hopa, murisi, ma?!  
        </mesaj>  
    </mesaje>  
    <mesaje lang="en">  
        <mesaj id="fail">  
            You failed.  
        </mesaj>  
        <mesaj id="castig">  
            You somehow won....  
        </mesaj>  
        <mesaj id="pierdut">  
            You died!  
        </mesaj>  
    </mesaje>  
</lang>

Pentru a incarca mesajele Loader-ul va avea o metoda Loader.parseMessages(). Aceasta poate primi limba ca parametru sau limba e setata in functia Loader.init() (metoda de initializare a loaderului). Loaderul va incarca doar mesajele limbii setate. Va avea o metoda Loader.getMessage(id) care va returna mesajul.

**Fisier xml de setari**

selecteaza textul<?xml version="1.0" encoding="utf-8" ?>  
<setari>  
    <nivele>  
        <incepator>  
            <nr\_adversari>10</nr\_adversari>  
            <viata\_adversari>100</viata\_adversari>  
            <viteza\_jucator>80</viteza\_jucator>  
        </incepator>  
        <mediu>  
            <nr\_adversari>20</nr\_adversari>  
            <viata\_adversari>200</viata\_adversari>  
            <viteza\_jucator>70</viteza\_jucator>  
        </mediu>  
        <avansat>  
            <nr\_adversari>30</nr\_adversari>  
            <viata\_adversari>300</viata\_adversari>  
            <viteza\_jucator>70</viteza\_jucator>  
        </avansat>  
    </nivele>  
    <gloante>  
        <glont id="g0">  
            <nume>glont de baza</nume>  
            <imagine>im\_g0</imagine><!-- id-ul imaginii corespunzatoare -->  
            <putere>10</putere>  
            <viteza>200</viteza>  
        </glont>  
        <glont id="g2">  
            <nume>glont mega</nume>  
            <imagine>im\_g1</imagine><!-- id-ul imaginii corespunzatoare -->  
            <putere>50</putere>  
            <viteza>100</viteza>  
        </glont>  
        <glont id="g3">  
            <nume>super glont</nume>  
            <imagine>im\_g2</imagine><!-- id-ul imaginii corespunzatoare -->  
            <putere>100</putere>  
            <viteza>300</viteza>  
        </glont>  
    </gloante>  
    <harti>  
        <jucator>j</jucator>  
        <turn>t</turn>  
        <zid>#</zid>  
        <liber>0</liber>  
        <harta id="h1">  
            <linie>000000t000######000000t</linie>  
            <linie>0000000000000##000##000</linie>  
            <linie>000#####000000000#####0</linie>  
            <linie>00000#000000t00000000#0</linie>  
            <linie>0j000#000#####00t0000#0</linie>  
            <linie>000000t0000000000#####0</linie>  
            <linie>00##00000###00000000000</linie>  
            <linie>00##0000000000t00000000</linie>  
        </harta>  
        <harta id="h2">  
            <linie>00t0000####00000000000t</linie>  
            <linie>0###00t000000##00000000</linie>  
            <linie>00000##############0000</linie>  
            <linie>00000#0000tttt0000#0000</linie>  
            <linie>0t00t#0t00tttt00t0#0000</linie>  
            <linie>0##00#t00000000000#0000</linie>  
            <linie>00#00###0#####0####0t00</linie>  
            <linie>j0#00000000000000000000</linie>  
        </harta>  
        ...  
    </harti>     ...  
</setari>

Fisierul de setari poate contine descrierile nivelelor, ale obiectelor din joc, pozitii initiale, harti predefinite.

**Namespace-ul jocului**

Jocul trebuie sa aiba propriul sau namespace, dar namespace-urile in Javascript sunt simbolizate prin obiecte. Deci cititi in laborator despre diversele moduri de definire a namespace-urilor si alegeti o metoda care sa se potriveasca bine cu ce aveti de facut.

**Fisierul de constante**

Ideal ar fi sa evitam hardcodarea valorilor. Daca scriem valorile direct, poate vrem apoi sa le modificam si trebuie sa cautam intr-un cod imens un numar (pe care poate nu-l mai tinem minte sau oricum si daca ni-l amintim, cu find mai fasim cateva pana dam de el). Astfel vom defini un fisier de constante, numit constante.js care va contine toate acele date care nu se potriveau prea bine in fisierele xml.

Cum fisierul de constante va fi folosit de catre modulul principal al jocului, va trebui inclus in html inaintea modulului principal. Deci fisierul de constante va fi cel care se va ocupa de crearea namespace-ului jocului in care va defini constantele.

Avem cel putin 2 variante de definire al lor. De mentionat este ca nu putem folosi const pentru ca nu putem creea proprietati constante ale unui obiect (cel putin nu in mod direct)

Daca obiectul corespunzator jocului il construim cu functia constructor sau ii definim o clasa prin *class*, atunci cel mai sigur mod e sa ne definim "constantele" ca variabile locale cu valori date, pentru care definim niste getteri.

Mai jos aveti exemplu pentru functia constructor:

selecteaza textulfunction Joc(){  
    var NR\_MILISECUNDE\_UPDATE=100;  
    var DEBUG\_MODE=true;//de exemplu o variabila care ne spune daca sa afisam anumite mesaje de debug  
    //etc.  
      
    this.getNrMilisUpdate=function()  
    {  
        return NR\_MILISECUNDE\_UPDATE;  
    }  
    this.getDebugMode()=function()  
    {  
        return DEBUG\_MODE;  
    }  
}

Putem realiza acelasi lucru si in stilul ES6 (dar nu va functiona pe browsere vechi):

selecteaza textulclass Joc{  
    //...  
    get NR\_MILISECUNDE\_UPDATE(){return 100;}  
    get DEBUG\_MODE(){return true;}  
    //...  
}

Sau putem defini obiectul cu object literal. Personal consider ca e mai buna aceasta varianta, deoarece clase definim pentru cazul in care dorim mai multe instante de acelasi tip. Dar in cazul nostru obiectul joc e unic.

selecteaza textuljoc=(function(){  
    var NR\_MILISECUNDE\_UPDATE=100;  
    var DEBUG\_MODE=true;  
    //...  
    return {  
            getNrMilisUpdate:function()  
            {  
                return NR\_MILISECUNDE\_UPDATE;  
            },  
            getDebugMode:function()  
            {  
                return DEBUG\_MODE;  
            }  
            //...  
        }  
    })();

**Clase custom pentru entitatile aplicatiei**

De exemplu, pentru o **aplicatie-joc**, sa ne imaginam ca avem pe ecran niste adversari cu care se lupta cu jucatorul. Fiecare adversar are niste caracteristici, de exemplu: viteza, viata etc. Fara sa grupam toate caractersticile intr-un obiect am ajunge sa avem vector de viteze, vector de vieti etc. Si sa recuperam caracteristicile cu ajutorul unui indice insa asta ar complica mult lucrurile. Cand vrem sa stergem un astfel de obiect, trebuie sa ii stergem proprietatile din toti acei vectori. Arm mai putea memora datele chiar in obiectul DOM asociat, creand proprietati noi. Insa daca faceti asta puteti avea probleme cu obiectele respective; puteti suprascrie din gresala o proprietate default sau browserul sa nu le trateze cum trebuie din cauza acelor proprietati in plus. Doar in dataset ar putea fi adaugate proprietati dar si aici suntem limitati de faptul ca dataset cuprinde doar proprietati cu valori String, deci nu am putea sa memoram referinte la obiecte, de exemplu. Asadar cea mai eleganta metoda si cea care ne ajuta sa evitam eventuale bug-uri este sa cream obiecte (fie prin clasa/functie constructor, fie prin object literal) pentru fiecare entitate a jocului: jucator, adversar, obstacol etc.

Daca insa pentru anumite elementele nu avem proprietati in plus fata de cele oricum oferite de interfata DOM, nu mai e necesar sa facem clase pentru ele. De exemplu daca un obstacol trebuie lovit de 3 ori ca sa dispara atunci avem proprietate in plus pentru ca trebuie sa stim de cte ori a fost lovit pana in momentul curent. Dar daca obstacolul doar exista pe ecran si jucatorul nu poate sa se deplaseze peste el, atunci singurele proprietati care ne intereseaza sunt pozitia si dimensiunea, iar aceastea oricum sunt memorate in stilul obiectului DOM, deci in acest caz nu ar mai fi necesara crearea unei clase pentru el.

Pentru o aplicatie de procesare a datelor din pagina, de exemplu una care proceseaza datele unor produse, am dori sa grupam toate acele date intr-un singur obiect.

Pentru aplicatii care schimba tema site-ului putem avea obiecte referitoare la datele afisate si setarile utilizatorului pentru tema respectiva.

**Detalii de implementare pentru cei care aleg sa faca o aplicatie de tip joc**

Pentru o structurare mai buna a jocului va propun urmatoarele idei de organizare si implementare.

**Ecranul de setari**

**Nu folositi trimiterea datelor prin submit pentru realizarea acestui task.** Valorile se vor prelua direct din inputuri. Veti avea un container cu minim urmatoarele tipuri de inputuri:

* input de tip range (de exemplu poate seta numarul de vieti, numarul de adversari, dimensiunea unui element pe ecran, numarul de secunde de game-play etc.)
* input de tip text (de exemplu pentru numele utilizatorului sau pentru un motto, sau pentru a scrie ceva intr-un anume format, cum ar fi un cheatcode)
* checkbox-uri (de exemplu pentru a seta daca sa apara ceva sau nu pe ecran, sau pentru cazul in care dorim sa avem valori multiple pentru o propritate, cum ar fi cazul in care avem mai multe tipuri de adversari din care putem alege oricate categorii)
* grup de radiobuttons (de exemplu pentru a seta nivelul de joc, pentru a alege o culoare de background, pentru a alege o anume harta, un anume tip de adversari, limba jocului, modul de calculare al scorului etc.)
* select simplu (idem radiobuttons)
* select multiplu (pentru valori multiple ale unui camp; vezi exemplul de la checkbox)
* textarea (poate fi folosit pentru ceva asemenator cu cazurile de la inputul de tip text, sau pentru selectarea unor valori multiple daca le separam printr-un caracter special, de exemplu, culorile adversarilor: "red;green;blue")

Tot in container va fi si un buton. La click pe acest buton se vor prelua datele din inputuri si jocul va porni cu setarile

Inputurile de tip text si textarea-urile vor avea si un placeholder care explica formatul datelor. Datele introduse de utilizator vor fi verificate cu ajutorul RegExp.

**Functia de update a jocului**

Uneori avem interactiuni intre elementele care se misca pe ecran. De exemplu vrem sa vedem coliziunea intre ele. Daca fiecare dintre ele are propria sa functie de miscare asociata (conform exemplului dat la laborator) atunci, daca verificam in cadrul functiei de miscare, detectam coliziunea de 2 ori pentru 2 obiecte care s-au intersectat. Cel mai simplu mod de a trata problema esta sa avem o functie joc.update(), care fi a fost apelata cu setInterval, fie se autopeleaza printr-un setTimeout si care se ocupa de toate calculele si repozitionarile obiectelor de pe pagina. Astfel ar putea verifica fara probleme coliziunile intr-un grup de n elemente (verificand intai primul cu cele de la al 2-lea incolo, apoi pe al doilea cu cele de la al 3-lea incolo si tot asa.

In plus uneori vrem sa avem si un cronometrru pentru joc (sa stim de cand a inceput jocul). Functia de update ar putea sa se ocupe usor si de aceasta functionalitate. Deci in general ar face toate calculele care tin de animatii si de trecerea timpului. Functia de update ar trebui apelata destul de des, daca se ocupa de animatii, de exemplu undeva intre 100-200 milisecunde. Daca in schimb se ocupa de elemente care nu trebuie actulizate foarte des, cum e cronometrul care s-ar actualiza oricum la fiecare secunda, e ok sa fie apelata si undeva intre 500-1000 milisecunde.

**Folosirea sprite-urilor**

Sprite-ul reprezinta o imagine mare cu mai multe sub-imagini in interiorul sau. In joc acestea se "decupeaza" (folosind proprietatea clip). Motivul folosirii unui sprite in locul mai multor imagini mici este ca un singur transfer cu imaginea mare poate dura mai putin decat toate transferurile imaginilor mici.

**Un debugger live**

Uneori vrem sa vedem cum se schimba valorile unor variabile in timp real, fara a pune breakpoints. Astfel am avea nevoie de un obiect MyDebugger cu metoda MyDebugger.setWatch(text,referinta). Iar cand avem in cod o variabila pe care vrem sa o urmarim, pur si simplu o transmitem ca parametru metodei setWatch() cu un text asociat care sa ne ajute sa identificam variabila in zona de afisare. Pentru variabilele urmarite ar trebui sa avem un tabel, sau macar un div cu randuri de forma text\_setat=valoare in care sa urmarim cum se schimba valorile.

**Resurse utile pentru aplicatie**

**Imagini gratuite:**

* <http://openclipart.org/homepage>
* <http://www.allfree-clipart.com/>
* <http://www.iconspedia.com/> - fisierele pot fi downloadate si ca png si le puteti folosi in joc.
* <http://www.freepik.com/free-photos-vectors/png> -aici puteti gasi si sprite-uri
* <http://www.incredibleart.org/links/clipart.html> - lista cu foarte multe siteuri de imagini gratuite

**Sprite-uri gratuite**

* <https://www.gameart2d.com/freebies.html>

**Cursoare gratuite**

* <http://www.rw-designer.com/cursor-library>

**Fisiere audio gratuite**

* <https://freesound.org/browse/> - folositi search box-ul pentru a cauta tipul de sunete dorit
* <http://www.noiseforfun.com/browse-sound-effects/> - aveti categoriile de sunete in dreapta

**Barem proiect**

**Baremul este extensibil, in sensul ca daca in urma discutiilor cu studentii decidem ca anumite elemente pe care oricum le-ar fi implemntat ar trebui punctate si ele, se pot adauga optiuni noi in barem. Baremul insa nu se va schimba in privinta elementelor deja existente.**

In barem sunt trecute denumirile proprietatilor si metodelor din DOM, dar se puncteaza si daca ati folosit echivalentele din jQuery

Cand lucrati la joc, alegeti din barem elementele pe care doriti sa le implementati. Punctajul total depaseste cu mult punctul alocat deci aveti o mare libertate de alegere. Si evident, se puncteaza si se aduna la nota si elementele care depasesc punctajul alocat.

Elementele din categoria functionalitati sunt cele care se puncteaza cel mai mult si in rezolvarea acestora sigur vor fi cuprinse mai multe elemente de sintaxa javascript.

**Elementele eliminatorii (cerintele minimale) trebuie implementate in totalitate.** Obiectivele **recomandate** sunt elemente foarte importante care apar des la examen. Elementele optionale (fara marcaj) se refera la elemente mai putin importante (tot pot sa apara la examen dar sunt incluse mai rar in subiecte) sau sunt functionalitati specifice unui anume timp de aplicatie (si deci nu se aplica tuturor proiectelor). Punctajele sunt intre un minim si un maxim. Punctajul minim se ofera pe un obiectiv implementat la nivel minim de complexitate sau daca e pus in aplicatie "doar ca sa fie" ( de exemplu un buton care la click afiseaza o alerta cu mesajul "ai dat click" va primi un punctaj minim). Punctajul maxim se obtine mai greu si se aplicat atunci cand obiectivul e la cote mari de complexitate si are rol important in proiect. In general studentii iau un punctaj undeva pe la mijloc pentru fiecare obiectiv (deci tineti cont de acest aspect daca doriti sa va faceti suma punctelor. Ideal e sa lucrati astfel incat sa depasiti un pic 1punctajul alocat proiectului ca sa fiti siguri ca obtineti nota maxima deoarece e posibil sa considerati o complexitate mai mare pentru un element si sa ii estimati un punctaj mai mare, sau sa va incurcati la prezentare si sa pierdeti punctele pe acel obiectiv).

**Modularizare si organizare proiect**

1.  (identificator: ajax; **recomandat**; punctaj minim:**0.02**; punctaj maxim:**0.1**;) incarcarea datelor dintr-un fisier XML/Json prin AJAX.

**Observatie:** se puncteaza toate utilizarile distincte (daca intr-adevar difera suficient ca functionalitate).

1.  (identificator: algoritm\_js; **recomandat**; punctaj minim:**0.05**; punctaj maxim:**0.3**;) algoritm in JavaScript (care calculeaza ceva special cum ar fi un drum in graf sau o dispunere optima a adversarilor pe harta in cadrul unui joc etc.). Functionalitatea in sine a aplicatiei si relevanta ei fata de tema proiectului..
2.  (identificator: fisier\_extern\_js; **eliminatoriu**; punctaj minim:**0.01**; punctaj maxim:**0.05**;) punerea codului JavaScript in fisiere externe. Punctajul e mai mare cu cat modularizarea e mai buna (se respecta structura cu fisiere separate pentru fiecare clasa, pentru fisierul de constante pentru scriptul general (main) etc). Nu trebuie sa avem cod java script scris in interiorul tagului script..
3.  (identificator: obiectul\_loader; **recomandat**; punctaj minim:**0.05**; punctaj maxim:**0.15**;) obiect care se ocupa de administrarea resurselor si setarilor din fisierele de date xml.
4.  (identificator: obiectul\_Debugger; **recomandat**; punctaj minim:**0.05**; punctaj maxim:**0.05**;) obiect care permite urmarirea variabilelor in mod live (afiandu-le valorile pe ecran si facand update asupra lor suficient de des).
5.  (identificator: cod\_corect\_ordonat; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.05**;) cod ordonat, variabile cu nume sugestive, comentarii. Sa nu existe erori in consola..
6.  (identificator: tratare\_erori; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.05**;) throw + try,catch.
7.  (identificator: namespace\_aplicatie; **recomandat**; punctaj minim:**0.02**; punctaj maxim:**0.05**;) Namespace-ul aplicatiei.
8.  (identificator: constantele\_aplicatiei; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.03**;) fisierul constante.js care contine constantele aplicatiei.
9.  (identificator: structura\_creata\_dinamic; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.15**;) "crearea dinamica a containerului aplicatiei si a elementelor din cadrul ei " (de exemplu desenati un soare cu ajutorul divurilor) aplicand culori, transformari, umbre si alte efecte. Crearea dinamica a altor ecrane speciale ale aplicatiei. Se puncteaza in functie de diversitatea si complexitatea elementelor generate. Crearea elementelor trebuie sa se realizeze intr-o functie de initializare..

**Observatie:** se puncteaza toate utilizarile distincte (daca intr-adevar difera suficient ca functionalitate).

1.  (identificator: impartire pe foldere; *optional*; punctaj minim:**0.01**; punctaj maxim:**0.04**;) Impartirea proiectului pe foldere (de exemplu, foldere de imagini, de videoclipuri, de sunete, de fisiere de stil, de fisiere js etc).
2.  (identificator: functie\_update; **recomandat**; punctaj minim:**0.03**; punctaj maxim:**0.1**;) functie de update a ecranului aplicatiei (in special pentru jocuri dar si pentru alte aplicatii cu continut schimbat in real time)..

**Afisare, aspect si accesibilitate**

1.  (identificator: afisare\_blocuri\_informative; **recomandat**; punctaj minim:**0.02**; punctaj maxim:**0.1**;) In urma diverselor evenimente din aplicatie. De exemplu, intr-o aplicatie stil joc, utilizatorul a omorat un adversar sau a gasit un item, sau s-a ciocnit de un obstacol etc. Intr-o aplicatie de calculare/filtrare mesaje care arata cate elemente au fost afisate sau s-au eliminat etc). Se vor afisa undeva pe ecran mesaje informative. Pentru un puctaj diferit de minim, nu folositi alert. Trebuie sa fie continut generat in pagina. Punctajul e afectat de numarul mesajelor dar mai ales de cazurile in care acestea apar..
2.  (identificator: tutorial; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.1**;) Implementarea unui tutorial. Prezentare interactiva, treptata a interfetei si continutului aplicatiei, marcand pe rand portiunile semnificative de pe ecran si explicand de fiecare data la ce foloseste fiecare. Se puncteaza in plus daca are si o optiune de skip..
3.  (identificator: loading\_screen; **recomandat**; punctaj minim:**0.03**; punctaj maxim:**0.08**;) exisenta unui loading screen (cat timp sunt incarcate resursele se va afisa un progress bar care creste trepat de la 0 la 100%). Pentru un punctaj diferit de minim veti lua o cerinta de progressbar custom, de la mine..
4.  (identificator: autosave; **recomandat**; punctaj minim:**0.02**; punctaj maxim:**0.02**;) salvare automata a unor date ale aplicatiei in localStorage. Pentru un joc poate salva intreaga stare a jocului pentru a se putea reveni la el. Pentru o aplicatie de calculare/filtrare se pot memora optiunile utilizatorului si ce era afisat pe ecran in momentul salvarii. In cazul in care se implementeaza aceasta functionalitate pe langa cea de salvare voluntara (declansata de utilizator), utilizatorul ar putea avea de ales in cazul in care intra pe o optiune "load state" intre salvarile sale si cea generata de autosave..
5.  (identificator: avertisment\_iesire\_pagina; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.04**;) Avertisment la iesirea de pe pagina. De exemplu, in caz ca utilizatorul nu si-a salvat activitatea de ceva timp, sau a depasit o anume etapa importanta si doreste sa paraseasca aplicatia fara a fi salvat ce a facut, este recomandat sa primeasca un mesaj la inchiderea paginii in care sa fie intrebat daca intr-adevar doreste sa paraseasca pagina, si in caz ca raspunde nu, sa nu se mai inchida fereastra..
6.  (identificator: aspect; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.1**;) Punctaj dat pe aspectul aplicatiei (cat de ingrijita e asezarea in pagina, cat de bine sunt alese imaginile etc)..
7.  (identificator: afisare\_instructiuni\_js; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.1**;) afisarea instructiunilor aplicatiei(de exemplu afisate in urma unui eveniment sau intr-un fel de slideshow creat prin javascript). Se puncteaza minim daca instructiunile sunt pur si simplu scrise static in html.
8.  (identificator: responsive; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.1**;) pagina jocului sa fie responsive.
9.  (identificator: reset\_aplicatie; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.05**;) resetare prin javascript a aplicatiei.

**Date**

Daca doriti, puteti lucra cu JSON in loc de xml.

1.  (identificator: date\_resurse; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.05**;) fisierul xml/json de resurse (depinde de cat de variate sunt resursele: imagini, sunete, videoclipuri, cursoare etc).
2.  (identificator: date\_mesaje; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.05**;) xml-ul/json-ul de mesaje in aplicatie. Punctajul depinde de numarul mesajelor si proprietatile lor, dar creste mai ales daca proiectul este multilingv (recomandat pentru 2 limbi: engleza si romana)..
3.  (identificator: date\_setari; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.05**;) fisierul xml/json cu setarile aplicatiei (depinde de numarul de setari si influenta lor asupra aplicatiei).

**Clase predefinite**

Tot ce tine de crearea obiectelor custom

1.  (identificator: Array; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.1**;) folosirea clasei Array (punctajul creste in functie de complexitate, numarul de proprietati si metode distincte folosite, relevanta lor in proiect).
2.  (identificator: String ; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.1**;) folosirea clasei String (punctajul creste in functie de complexitate, numarul de proprietati si metode distincte folosite, relevanta lor in proiect).
3.  (identificator: Math; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.1**;) folosirea clasei Math (punctajul creste in functie de complexitate, numarul de proprietati si metode distincte folosite, relevanta lor in proiect).
4.  (identificator: elemente random; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.05**;) generarea unor proprietati aleatoare(culori, pozitii, tipuri de imagini etc.) folosind metoda random din clasa Math.
5.  (identificator: Date ; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.1**;) folosirea clasei Date (punctajul creste in functie de complexitate, numarul de proprietati si metode distincte folosite, relevanta lor in proiect).
6.  (identificator: RegExp; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.1**;) verificarea datelor introduse de un utilizator prin RegExp (punctajul creste in functie de complexitate, numarul de proprietati si metode distincte folosite, relevanta lor in proiect).

**Obiecte custom**

Tot ce tine de crearea obiectelor custom

1.  (identificator: obiecte\_newObject; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.05**;) crearea de obiecte custom prin varianta new Object() sau cu {} si adaugarea de proprietati si metode.
2.  (identificator: obiecte\_custom; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.3**;) Crearea de obiecte custom prin varianta cu functie constructor sau cu class. Punctajul poate creste usor daca sunt mai multe clase relevante implementate pentru aplicatie. Punctajul e cel mai afectat de complexitatea claselor:
   * posibilitatea de a apela constructorul (sau alte metode) cu numar diferit de parametri. Valori default pentru unii parametri.
   * Tipurile proprietatilor: private, publice, statice.
   * Tipurile metodelor: private, publice, statice
   * Definirea unor metode getter si setter
   * definirea proprietatilor si metodelor folosind prototype. Apelul extern al unei functii definite prin prototype.
   * Implementarea unor clase derivate (mostenire)

.

**DOM - general**

1.  (identificator: creare dinamica elemente DOM; **eliminatoriu**; punctaj minim:**0.01**; punctaj maxim:**0.04**;) creare dinamica element DOM (de exemplu prin document.createElement sau prin clonare cloneNode) si adaugare in pagina.
2.  (identificator: creare dinamica TextNode; **eliminatoriu**; punctaj minim:**0.01**; punctaj maxim:**0.01**;) creare dinamica textNode (de exemplu prin document.createTextNode) si adaugare in pagina.
3.  (identificator: adaugare dinamica DOM; **eliminatoriu**; punctaj minim:**0.01**; punctaj maxim:**0.05**;) adaugare dinamica element DOM cu appendChild si insertBefore.
4.  (identificator: stergere dinamica DOM; **eliminatoriu**; punctaj minim:**0.01**; punctaj maxim:**0.03**;) stergere dinamica element DOM (de exemplu cu removeChild).
5.  (identificator: inlocuire dinamica element DOM; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.02**;) de exemplu prin replaceChild().
6.  (identificator: actualizare element DOM; **eliminatoriu**; punctaj minim:**0.01**; punctaj maxim:**0.07**;) folosirea proprietatii innerHTML pentru elemente si a lui nodeValue pentru textNodes, setarea in mod dinamic a atributelor elementelor.
7.  (identificator: prelucrare tip nod; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.03**;) folosirea proprietatilor nodeType si tagName.
8.  (identificator: selectare elemente DOM; **eliminatoriu**; punctaj minim:**0.01**; punctaj maxim:**0.15**;) Manevrare DOM. Folosirea metodelor getElementById, getElementsByName, getElementsByTagName, getElementsByClassName, querySelectorAll, querySelector.
9.  (identificator: navigare DOM; **eliminatoriu**; punctaj minim:**0.01**; punctaj maxim:**0.15**;) Manipulare DOM. Deplasarea prin arborele DOM folosind proprietatile specifice: children, childNodes, firstChild, firstElementChild, lastChild, lastElementChild, parentNode/parentElement, nextSibling, nextElementSibling, previousSibling, previousElementSibling). Punctajul variaza in functie de numarul de proprietati distincte folosite dar si de relevanta in aplicatie..
10.  (identificator: folosirea si actualizarea dinamica a claselor elementelor; **recomandat**; punctaj minim:**0.02**; punctaj maxim:**0.07**;) folosirea proprietatilor className si classList (impreuna cu metodele lui classList -> mai multe metode distincte = punctaj mai mare).
11.  (identificator: schimbare\_css; **eliminatoriu**; punctaj minim:**0.01**; punctaj maxim:**0.1**;) Setarea unor proprietati CSS prin JavaScript (punctajul depinde de numarul proprietatilor distincte dar si de relevanta lor in aplicatie). Eventual schimbarea starii unui element in functie de un eveniment sau de rolul lui in joc( de exemplu apare un border pe un element selectat; se schimba imaginea adversarului daca acesta se pregateste de atac).

**Media**

1.  (identificator: elem\_audio; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.05**;) folosirea a cel putin unui element audio incarcat dinamic (prin JS).
2.  (identificator: elem\_video; *optional*; punctaj minim:**0.01**; punctaj maxim:**0.05**;) folosirea a cel putin unui element video incarcat dinamic (prin JS).
3.  (identificator: sunet\_asociat\_eveniment; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.05**;) sunet asociat unui eveniment (de exemplu la o coliziune, sau la click, sau la miscarea jucatorului etc). Punctajul se da si in functie de modul de detctare al evenimentului..

**Observatie:** se puncteaza toate utilizarile distincte (daca intr-adevar difera suficient ca functionalitate).

**Evenimente**

1.  (identificator: eveniment\_mouse\_coord\_win; **eliminatoriu**; punctaj minim:**0.02**; punctaj maxim:**0.06**;) preluare coordonate mouse in fereastra/document sau fata de un element DOM din body.
2.  (identificator: eveniment\_onload; **eliminatoriu**; punctaj minim:**0.02**; punctaj maxim:**0.1**;) folosirea evenimentului load (in special pentru ce se intampla la incarcarea paginii). Punctajul e afectat si numarul de actiuni efectuate la incarcare dar si de relevanta lor in proiect.
3.  (identificator: eveniment\_onchange; **recomandat**; punctaj minim:**0.02**; punctaj maxim:**0.1**;) evenimentele specifice inputurilor: change, focus, blur.
4.  (identificator: eveniment\_ctrl\_alt\_shift; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.03**;) verificarea faptului ca una din tastele ctrl, alt sau sfift e apasata cand se petrece un eveniment - nota complexitate consta in cate dintre cele 3 taste speciale au fost verificate in proiect; de exemplu pentru ctrl si shift, nota complexitate e 2).
5.  (identificator: eveniment\_capturing ; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.03**;) adaugarea unui eveniment a carui transmitere sa fie captata in faza de capturing.
6.  (identificator: eveniment\_stopPropagation; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.01**;) oprirea bubblingului.
7.  (identificator: eveniment\_mouse; **eliminatoriu**; punctaj minim:**0.02**; punctaj maxim:**0.2**;) folosirea unui eveniment de mouse. Creste punctajul daca evenimentele sunt adaugate dinamic si prin metode diferite: prin proprietati de tip on+nume\_eveniment sau cu metoda addEventListener(). Punctajul depinde de numarul evenimentelor distincte folosite dar si de relevanta lor in aplicatie: mouseup, mousedown, click, dblclick, mouseover, mouseout, mouseenter, mouseleave, mousemove + evenimente de *drag&drop*: drag, dragstart, dragend, dragenter,dragover, dragleave, drop.
8.  (identificator: eveniment\_tastatura; **eliminatoriu**; punctaj minim:**0.02**; punctaj maxim:**0.1**;) folosirea unui eveniment de tastatura. Creste punctajul daca evenimentele sunt adaugate dinamic si prin metode diferite: prin proprietati de tip on+nume\_eveniment sau cu metoda addEventListener(). Punctajul depinde de numarul evenimentelor distincte folosite dar si de relevanta lor in aplicatie: keypress, keyup, keydown.
9.  (identificator: preventDefault; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.03**;) folosirea metodei preventDefault (undeva unde este nevoie, nu doar ca sa fie).
10.  (identificator: loadedData; **recomandat**; punctaj minim:**0.02**; punctaj maxim:**0.02**;) Folosirea unui eveniment loadedData.
11.  (identificator: evenimente keyframes; **recomandat**; punctaj minim:**0.02**; punctaj maxim:**0.12**;) folosirea evenimentelor referitoare la animatii (keyframes): AnimationStart, AnimationIteration, AnimationEnd.

**Observatie:** se puncteaza toate utilizarile distincte (daca intr-adevar difera suficient ca functionalitate).

**Inputuri**

Atentie, inputurile se vor puncta numai daca se preia valoarea lor in javascript (fara a folosi transmiterea de date a unui formular prin metoda get). Regula se aplica tuturor tipurilor de input, select, textarea, cu exceptia inputurilor de tip submit, reset si button.

1.  (identificator: input\_range; **eliminatoriu**; punctaj minim:**0.01**; punctaj maxim:**0.07**;) input de tip range. Pentru punctaj maxim se creeaza elementul in mod dinamic si se preia valoarea lui in JS.
2.  (identificator: input\_text; **eliminatoriu**; punctaj minim:**0.01**; punctaj maxim:**0.05**;) input de tip text. Pentru punctaj maxim se creeaza elementul in mod dinamic si se preia valoarea lui in JS.
3.  (identificator: input\_checkbox; **eliminatoriu**; punctaj minim:**0.01**; punctaj maxim:**0.05**;) input de tip checkbox. Pentru punctaj maxim se creeaza elementul in mod dinamic si se preia valoarea lui in JS.
4.  (identificator: input\_radio; **eliminatoriu**; punctaj minim:**0.01**; punctaj maxim:**0.07**;) input de tip radio. Pentru punctaj maxim se creeaza elementul in mod dinamic si se preia valoarea lui in JS.
5.  (identificator: select\_simplu; **eliminatoriu**; punctaj minim:**0.02**; punctaj maxim:**0.08**;) folosirea unui select simplu. Pentru punctaj maxim se creeaza elementul in mod dinamic si se preia valoarea lui in JS.
6.  (identificator: select\_multiple; **recomandat**; punctaj minim:**0.02**; punctaj maxim:**0.1**;) folosirea unui select cu atributul multiple. Pentru punctaj maxim se creeaza elementul in mod dinamic si se preia valoarea lui in JS.
7.  (identificator: textarea; **eliminatoriu**; punctaj minim:**0.01**; punctaj maxim:**0.05**;) folosirea unui textarea. Pentru punctaj maxim se creeaza elementul in mod dinamic si se preia valoarea lui in JS.
8.  (identificator: input\_disabled ; **recomandat**; punctaj minim:**0.005**; punctaj maxim:**0.01**;) folosirea unui input care isi schimba starea din disabled in enabled sau invers). Pentru punctaj maxim se creeaza elementul in mod dinamic si se preia valoarea lui.
9.  (identificator: tag\_datalist; *optional*; punctaj minim:**0.01**; punctaj maxim:**0.05**;) Folosirea unui datalist asociat unui input. Pentru punctaj maxim se creeaza elementul in mod dinamic si se preia valoarea lui in JS.

**Obiectul window**

1.  (identificator: eveniment\_onresize; **recomandat**; punctaj minim:**0.02**; punctaj maxim:**0.05**;) folosirea evenimentului resize. Se puncteaza in functie de functionalitatile implementate la resize..
2.  (identificator: window.setTimeout; **eliminatoriu**; punctaj minim:**0.01**; punctaj maxim:**0.1**;) folosirea functiei setTimeout impreuna cu functia de anulare a timeout-ului: clearTimeout. Se puncteaza in functie de numarul de functionalitati implementate si complexitatea lor..

**Observatie:** se puncteaza toate utilizarile distincte (daca intr-adevar difera suficient ca functionalitate).

1.  (identificator: window.setInterval; **eliminatoriu**; punctaj minim:**0.01**; punctaj maxim:**0.1**;) folosirea functiei setInterval impreuna cu functia de anulare a intervalului: clearInterval. Se puncteaza in functie de numarul de functionalitati implementate si complexitatea lor..

**Observatie:** se puncteaza toate utilizarile distincte (daca intr-adevar difera suficient ca functionalitate).

1.  (identificator: window.getComputedStyle; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.01**;) folosirea metodei getComputedStyle.
2.  (identificator: alerte; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.05**;) folosirea unui alert/confirm/prompt.
3.  (identificator: window.open; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.1**;) crearea dinamica a unei ferestre si scrierea in cadrul acesteia (se puncteaza in functie de complexitatea contentului generat de voi prin JS in noua fereastra).

**Observatie:** se puncteaza toate utilizarile distincte (daca intr-adevar difera suficient ca functionalitate).

1.  (identificator: window.innerWidth\_innerHeight; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.01**;) folosirea proprietatilor innerWidth innerHeight ale ferestrei.

**Formulare**

1.  (identificator: input\_submit; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.03**;) input de tip submit.
2.  (identificator: input\_reset; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.03**;) input de tip reset.
3.  (identificator: input\_preluareDateForm; **recomandat**; punctaj minim:**0.05**; punctaj maxim:**0.05**;) preluarea valorilor dintr-un formular care a transmis datele cu metoda GET.
4.  (identificator: eveniment\_onsubmit; *optional*; punctaj minim:**0.02**; punctaj maxim:**0.05**;) folosirea evenimentului submit.
5.  (identificator: eveniment\_onreset; *optional*; punctaj minim:**0.02**; punctaj maxim:**0.05**;) folosirea evenimentului reset.
6.  (identificator: tag\_output; *optional*; punctaj minim:**0.01**; punctaj maxim:**0.03**;) Folosirea unui tag output si popularea lui cu date prin JS..

**Storage**

1.  (identificator: localStorage; **recomandat**; punctaj minim:**0.02**; punctaj maxim:**0.15**;) folosirea metodelor si proprietatilor specifice localStorage (eventual si sessionStorage): setItem, getItem, removeItem, clear etc..

**Diverse**

1.  (identificator: canvas; *optional*; punctaj minim:**0.03**; punctaj maxim:**0.3**;) folosirea unui element canvas (nota de complexitate depinde de elementele desenate, functionalitatile implementate si eventuale animatii in canvas).
2.  (identificator: svg; *optional*; punctaj minim:**0.02**; punctaj maxim:**0.1**;) manipularea proprietatilor unui element SVG prin JS.
3.  (identificator: jQuery; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.1**;) folosirea jQuery undeva in proiect (cu cat e mai mult cod scris in jQuery cu atat nota de complexitate va fi mai mare).
4.  (identificator: framework; *optional*; punctaj minim:**0.01**; punctaj maxim:**0.2**;) Folosirea altor framework-uri de JavaScript in afara de jQuery (Prototype, Moo Tools etc). Atentie, folosirea unui framework nu inseamna doar sa includeti scriptul, ci efectiv sa folositi functii de acolo). Ideal insa este sa folositi framework-uri numai acolo unde e neaparat necesar si in rest sa rezolvati totul folosind interfata DOM sau metodele oferie de jQuery - deoarece acestea sunt cele cerute la examen..
5.  (identificator: plugin\_jQuery; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.1**;) [Crearea unui plugin de jQuery](https://learn.jquery.com/plugins/basic-plugin-creation/" \t "_blank). Se puncteaza numai daca pluginul e folosit undeva pe site. Se va puncta fiecare plugin distinct (exceptie facand cazul cand sunt foarte asemanatoare: de exemplu - un plugin care aplica o anume culoare de background si un plugin care pur si simplu aplica alta culoare).

**Observatie:** se puncteaza toate utilizarile distincte (daca intr-adevar difera suficient ca functionalitate).

1.  (identificator: cookie-uri; *optional*; punctaj minim:**0.01**; punctaj maxim:**0.05**;) setarea cookie-urilor, obtinerea valorilor din ele, stergerea cookie-urilor.
2.  (identificator: test\_coliziune; **recomandat**; punctaj minim:**0.02**; punctaj maxim:**0.1**;) testarea coliziunii (a suprapunerii dintre doua elemente).
3.  (identificator: sprite\_uri; **recomandat**; punctaj minim:**0.03**; punctaj maxim:**0.03**;) Folosirea sprite-urilor pentru imagini. Se va puncta decuparea imaginilor din sprite-urile respective. Desi e optional e bine de implemnetat deoarece creste performanta aplicatiei..
4.  (identificator: animatie\_background; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.03**;) Animatie de background facuta in JavaScript.
5.  (identificator: ciocnire\_elastica; **recomandat**; punctaj minim:**0.02**; punctaj maxim:**0.02**;) Simularea unei ciocniri elastice intre doua elemente (de exemplu pentru o minge care se loveste de un perete).
6.  (identificator: scor; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.05**;) calcularea unui scor in functie de actiunile jucatorului.
7.  (identificator: documentatie; *optional*; punctaj minim:**0.01**; punctaj maxim:**0.05**;) O documentatie a proiectului scrisa in format html (cu descrierea temei, explicarea eventualilor algoritmi si listarea functionalitatilor majore implementate). Documentatia trebuie sa fie scurta si la obiect..
8.  (identificator: descrierea clara a aplicatiei; **eliminatoriu**; punctaj minim:**0.01**; punctaj maxim:**0.05**;) alegerea temei de proiect si comunicarea acesteia prin completarea formularului.

**Functionalitati**

1.  (identificator: animare\_obiect; **eliminatoriu**; punctaj minim:**0.05**; punctaj maxim:**0.2**;) schimbarea animata (prin javascript) a unei proprietati de tip coordonate sau dimensiuni ale unui obiect. De exemplu, deplasarea unui obiect:
   * printr-o animatie independenta de actiunile utilizatorului (realizata prin JavaScript, nu prin CSS)
   * in urma unui eveniment provocat de utilizator. Modificarile de tip diferit se puncteaza separat.

Obiectul se va muta treptat (nu brusc), de exemplu pe un segment, de la un punct P1 la un punct P2 pe ecran, sau pe un cerc sau pe orice alta traiectorie. Nu e obligatoriu sa folositi keyframes. Animatii pe traiectorii de tipuri diferite se puncteaza separat (de exemplu, nu se vor puncta separat doua cazuri in care obiectul se deplaseaza pe un segment. Dar daca o data se deplaseaza pe segment, o data pe cerc, atunci se puncteaza separat). Punctajul depinde si de complexitatea traiectoriei..

1.  (identificator: highscores; **recomandat**; punctaj minim:**0.05**; punctaj maxim:**0.05**;) Retinerea scorurilor in localStorage si afisarea unei liste de highscores..
2.  (identificator: save\_load\_state; **recomandat**; punctaj minim:**0.07**; punctaj maxim:**0.07**;) Optiunea de a salva starea aplicatiei(in localStorage) si de a continua utilizarea aplicatiei mai tarziu, reluand activitatea de unde a fost lasata (reincarcarea aplicatiei).
3.  (identificator: ecran\_pauza; **recomandat**; punctaj minim:**0.04**; punctaj maxim:**0.1**;) Optiunea de pauza a jocului. In special daca jocul presupune o activitate intensa din partea utilizatorului, de exemplu e un joc in care tot vin inamici spre personaj, este bine sa existe optiunea de oprire temporara a jocului, pentru ca utilizatorul sa poata lua o pauza..
4.  (identificator: undo\_list; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.05**;) Undo list. In cazul in care jocul permite acest lucru, veti implementa optiunile de undo si redo, in cazul in care utilizatorul doreste sa anuleze o decizie anterioara sau sa revina la ea..
5.  (identificator: achievements; **recomandat**; punctaj minim:**0.05**; punctaj maxim:**0.05**;) Implementarea unui set de achievements. Trebuie sa fie minim 3. Achievements-urile se vor accesa apasand de exemplu un buton. Se pot afisa intr-o fereastra noua sau in containerul jocului..
6.  (identificator: statistici\_live; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.1**;) afisare statistici live (de exemplu un bloc informativ cu cati adversari sunt pe ecran la un moment dat, cat s-a deplasat jucatorul, cum i se schimba sanatatea si numarul de vieti).
7.  (identificator: game\_won\_lost; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.05**;) Implementarea starii de joc castigat/pierdut(existenta unui test pentru acest aspect si afisarea prin JS a mesajului corespunzator).
8.  (identificator: timer; **recomandat**; punctaj minim:**0.03**; punctaj maxim:**0.1**;) crearea unui cronometru sau temporizator care sa se actulizeze dinamic cu trecerea timpului.

**Observatie:** se puncteaza toate utilizarile distincte (daca intr-adevar difera suficient ca functionalitate).

1.  (identificator: cheatcodes; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.1**;) cheatcodes.

**Node**

1.  (identificator: login\_logout; **eliminatoriu**; punctaj minim:**0.05**; punctaj maxim:**0.1**;) Crearea unei sesiuni. Utilizatorul sa se poata loga/deloga. In cazul logarii, trebuie sa existe un afisaj pe fiecare pagina a site-ului in care sa se vada informatii despre userul logat. Se ofera punctaj minim pentru hardcodarea userului si parolei. Punctajul creste in cazul in care: se foloseste un fisier text( de exemplu xml sau json) ori o baza de date cu informatii despre utilizatori (username, parola etc), parola este criptata, se blocheaza login-ul pentru un anumit timp in cazul a N incercari esuate de login ale aceluiasi utilizator..
2.  (identificator: template\_ejs; **eliminatoriu**; punctaj minim:**0.03**; punctaj maxim:**0.15**;) Sa se foloseasca ejs. De exemplu: includerea headerului si a footerului, folosirea de template pentru zone de html repetitive (tabele cu date, articole pentru produsele site-ului, afisarea conditionala a anumitor sectiuni din site - de exemplu cand userul e logat etc.). Cu cat se folosesc mai multe elemente de ejs, cu atat punctajul e mai mare.
3.  (identificator: citire\_scriere\_date; **eliminatoriu**; punctaj minim:**0.04**; punctaj maxim:**0.15**;) Sa se citeasca data dintr-un fisier de date, prin node, si acele date sa fie folosite in pagini (de exemplu, se citesc produsele promovate de site dintr-un json si sunt afisate informatii despre ele intr-o pagina). De asemenea, sa se si scrie date in fisier, sau sa se actualizeze fisierul (de exemplu, in urma completarii unui formular de introducere a produselor noi). .
4.  (identificator: pagina\_404; **eliminatoriu**; punctaj minim:**0.02**; punctaj maxim:**0.05**;) Se va crea o pagina de eroare 404 (la care se ajunge daca s-a trecut in bara de adrese un link inexistent din site). Punctajul depinde de aspectul si relevanta paginii, dar si de eventualele cazuri implementate (afisaje diferite in pagina in functie de cerere).
5.  (identificator: preluare\_date\_formular; **eliminatoriu**; punctaj minim:**0.03**; punctaj maxim:**0.1**;) Preluarea datelor dintr-un formular completat de utilizator si procesarea datelor respective de catre server. Punctajul va creste daca se fac: verificari ale datelor, se dau raspunsuri custom in functie de date, datele sunt folosite pentru a genera o pagina in functie de valorile din formular, se leaga de actiuni suplimentare precum trimiterea unui mail sau a unui mesaj pe socket etc). Pentru a se primi punctajul, datele preluate din formular trebuie sa fie folosite cu un scop clar in cadrul aplicatiei..
6.  (identificator: redirectionare\_server; **recomandat**; punctaj minim:**0.01**; punctaj maxim:**0.03**;) Se va realiza un redirect pe partea de server. Se da punctaj mai mare daca redirectionarea are si niste conditii in spate (de exemplu un utilizator neautorizat intra pe o pagina interzisa).
7.  (identificator: trimitere\_mail; **recomandat**; punctaj minim:**0.02**; punctaj maxim:**0.04**;) Se va trimite un mail de la server in urma unei actiuni, de exemplu in urma inregistrarii pe site. Se da punctaj mai mare daca in mailuri diferite avem continuturi diferite (care pot depinde de datele dintr-un formular sau datele utilizatorului logat).
8.  (identificator: cookieuri\_server; **recomandat**; punctaj minim:**0.02**; punctaj maxim:**0.05**;) Setarea de cookieuri folosind node. punctajul se da in functie de complexitatea cookie-ului si conditiile de setare.
9.  (identificator: socket\_io; **recomandat**; punctaj minim:**0.05**; punctaj maxim:**0.15**;) Folosirea de sockets pentru un update al paginii provocat de catre server (de exemplu utilizatorul primeste o notificare de la server, sau un mesaj pe un chat etc). Punctajul se da in functie de complexitatea comunicarii, numarul de evenimente diferite folosite (tipuri diferite de notificari) etc..
10.  (identificator: upload\_fisiere; **recomandat**; punctaj minim:**0.02**; punctaj maxim:**0.1**;) se poate realiza upload de fisiere pe server. Se da punctaj mai mare daca fisierele pot fi afisate, sau sunt procesate de catre server (de exemplu se preiau date din ele, se proceseaza o imagine etc)..
11.  (identificator: conectare\_baza\_date; *optional*; punctaj minim:**0.04**; punctaj maxim:**0.1**;) Folosirea unei baze de date pentru stocarea datelor site-ului. Punctajul creste in numarul de query-uri de tip diferit, si in functie de complexitatea acestora..
12.  (identificator: publicare\_net; *optional*; punctaj minim:**0.04**; punctaj maxim:**0.06**;) Punerea aplicatiei node pe net, si verificarea tuturor functionalitatilor care mergeau local..
13.  (identificator: diverse\_node; *optional*; punctaj minim:**0**; punctaj maxim:**0.1**;) Orice nu e specificat in barem dar e rezolvat in node.

**Punctaj calculat cu nota minima de complexitate:**

**Mod de prezentare si notare**

Prezentarea se va face in ultimele doua saptamani. Daca cineva e gata mai devreme poate prezenta si inainte (insa nu in timpul laboratorului ci intr-un interval stabilit de comun acord pentru a nu rapi din timpul de predare).

Atentie, trebuie sa va cunoasteti proiectul. Nu se puncteaza nimic din ceea ce nu anuntati ca ati lucrat sau nu stiti sa prezentati. Chiar daca un obiectiv a fost implementat, daca nu e evidentiat de catre student la prezentare, nu e trecut in punctaj.

Inainte de prezentare studentul trebuie sa bifeze in barem ceea ce a lucrat. Prezentarea nu incepe pana studentul nu termina de bifat. Aceasta regula e necesara pentru a rezolva urmatoarele probleme:

* la prezentare ii va fi mai usor sa spuna ce a facut, fara sa caute in cod daca a folosit cutare sau cutare chestiune din barem. Timpul necesar prezentarii poate fi astfel redus.
* evitam cazul in care studentul uita sa prezinte vreunul dintre obiectivele implementate.
* Poate studentul a folosit o portiune din exemplele de la curs/laborator insa nu se simte prea sigur sa prezinte acea bucata. In mod normal nu ar trebui sa se intample asa ceva, insa daca respectivul cod e vital ca sa mearga o anume portiune din proiect, studentul poate opta sa nu il adauge in cerintele realizate, si astfel sa nu fie punctat pentru codul respectiv dar nici sa nu primeasca intrebari de acolo care ar putea micsora nota sa de prezentare.

**Atentie! Nota proiectului se va calcula astfel: grad\_respectare\_al enuntului\_scris\_pe\_site\*punctaj\_proiect\*nota\_prezentare/10.**. Deci daca ati luat 1p pe proiect in urma notarii elementelor din barem insa la prezentare nu stiti sa explicati aproape nimic, si luati sa zicem nota 1 sau 2, veti primi doar 0.1-0.2 puncte. Motivul pentru aceasta restrictie este unul evident: prezentarea e dovada ca studentul respectiv a facut el insusi tot ceea ce exista in proiectul cu care a venit. De acee se poate lua chiar si 0 pe prezentare in cazul in care studentul nu poate explica absolut nimic din proiect (cu alte cuvinte anularea proiectului).

Pentru calcularea notei de prezentare, studentul va fi pus sa explice portiuni de cod sau sa raspunda la intrebari.

Prezentarea proiectului trebuie sa dureze intre 10-15 minute deci daca studentului ii ia mai mult de jumatate de minut sa se gandeasca pana a raspunde la o intrebare se va considera ca nu stie si se va trece la urmatoarea intrebare.

[sus](http://irinaciocan.ro/tehnici_web/barem_proiect_js.php)

Copyright © 2010-prezent Irina Ciocan (irinaciocan.ro, e-mail: irina.ciocan@gmail.com)