ISBN 978-973-666-785-5

Doru E. TILIUȚE

HTML

CSS

Editura Univesității Ștefan cel Mare

2023

Doru E. TILIUȚE

Editare în limbaje pentru Web

ISBN 978-973-666-785-5

Editura Universităţii „Ştefan cel Mare”

2023

Pagină lăsată intenționat liberă

Internet și Web

De mai bine de două decenii viața noastră este conectată cu una dintre cele mai spectaculoase și revoluționare invenții ale secolului XX și, anume, Internetul. Comunicăm, ne informăm, schimbăm fișiere, actualizăm aplicațiile de pe telefon sau calculator, facem tranzacții bancare, rezervăm bilete de avion și multe altele prin intermediul Internetului. Dar ce este Internetul? Numele provine din asocierea cuvintelor în limba engleză Inter (între) și Nets (rețele) și desemnează o un sistem de comunicații între rețelele de calculatoare răspândite pe toată suprafața globului. Prin calculatoare înțelegem orice sistem de calcul, de la supercomputere la telefoanele inteligente și electrocasnicele inteligente. Rețeaua este un sistem de comunicații compus din echipamente de calcul (calculatoare sau dispozitive inteligente), interconectate și apte să schimbe date între ele. Prin urmare, o rețea de calculatoare conectează calculatoarele între ele iar Internetul conectează rețelele între ele.

Inițial, rețeaua Internet a aparținut Departamentului apărării al SUA și se numea ARPANET (Advanced Research Projects Agency Network). (Featherly). Termenul Internet a fost introdus în 1983, când rețeaua ARPANET s-a divizat în MILNET (Military Net) pentru uzul armatei și partea de uz civil al ARPANET.

Pentru a putea comunica, calculatoarele au nevoie de un sistem de adresare, prin care fiecărui să i se atribuie o adresă unică și un protocol de comunicație.

În Internet, sistemul de adrese este format din adresele IP (Internet Protocol), care identifică în mod unic orice echipament conectat la Internet. Calculatoarele conectate la Internet se numesc host-uri (host - gazdă). Așa cum fiecare clădire dintr-un oraș este identificată printr-o adresă la care să poată fi livrată corespondența, tot așa și host-urile dispun de o adresă la care să fie livrate mesajele electronice.

Adresele IP sunt astfel concepute încât să conțină atât adresa rețelei din care face parte un anumit calculator, cât și a host-ului în rețea. Prin analogie cu adresele clădirilor, putem avea aceeași stradă și același număr de clădire în două orașe diferite. Nu se poate crea nicio confuzie între cele două adrese atât timp cât ele conțin numele orașului (echivalent numărului sau numelui rețelei) și numărul străzii și al clădirii (echivalent numărului sau numelui calculatorului din rețea). Putem avea, așadar, două sau mai multe host-uri cu cu același număr, dar nu în aceeași rețea ci în rețele diferite. Pentru că adresele IP conțin ambele numere, de rețea și de *host*, adresa IP a oricărui calculator este unică. Situația este redată în Figura 1.

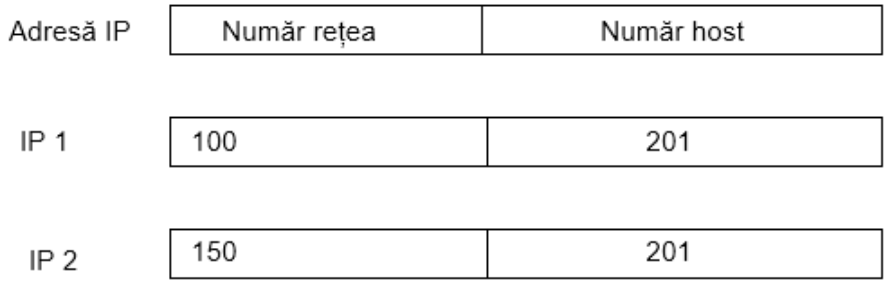


Figura 1. Două adrese IP diferite, în care există host-uri cu același număr. Diferența o face numărul rețelei

Protocolul este un ansamblu de reguli după care se face schimbul de date între echipamente. În Internet, protocol utilizat se numește TCP/IP (Transport Control Protocol/ Internet Protocol).

Rețelele de calculatoare pot utiliza și alte tipuri de adrese de identificare și alte protocoale de comunicare, dar Internetul reușește să interconecteze toate tipurile de rețele, indiferent de tehnologiile folosite în interiorul lor, ceea ce este cu totul și cu totul remarcabil.

De la inventarea lui, Internetul a oferit o serie de servicii și anume (Figura 2):

- FTP - File Transfer Protocol,
- Gopher - căutare, extragere și afișare de informații de pe alte calculatoare din Internet,
- Mail - trimitere de mesaje prin Internet,
- Newsgroup - grupuri de discuții,
- Telnet - conectare la distanță la un calculator conectat la Internet.

Au mai fost și altele, dar utilizarea lor a scăzut în timp pentru că au fost înlocuite de alte protocoale sau aplicații. Astfel, serviciul Telnet a fost înlocuit de SSH (Secure SHell Protocol), care permite o comunicație securizată între clientul conectat și calculatorul aflat la distanță. Newsgroup a fost înlocuit și el de forumuri, mult mai ușor de utiliza, Mail a fost înlocuit de e-mail (sau email). Alte servicii precum FTP și e-mail sunt încă folosite pe scară largă, cu versiuni securizate.

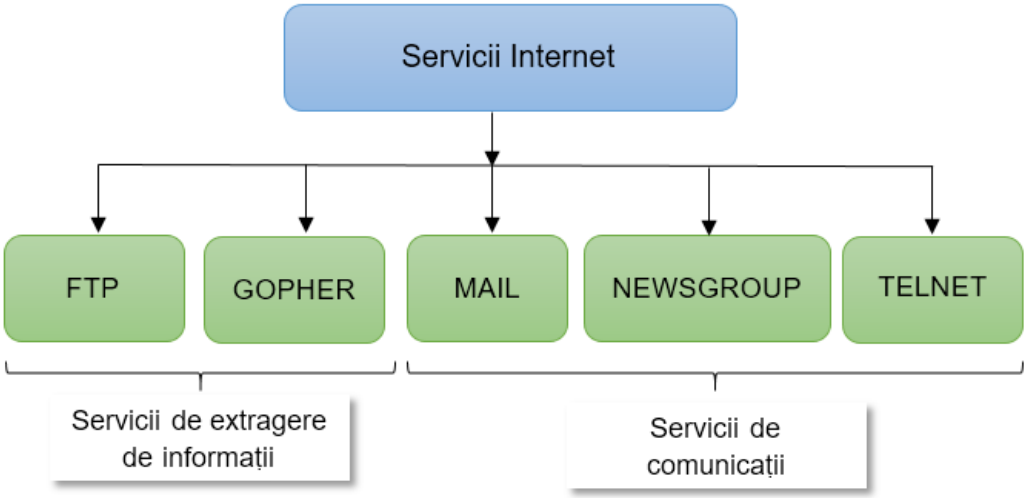


Figura 2. Câteva servicii ale Internetului

Toate serviciile Internet sunt de tipul „client-server”. Aceasta este o arhitectură în care programe de tip „client” sunt instalate pe o serie de calculatoare care emit cereri către alte calculatoare pe care sunt instalate programe de tip „server”. Serverele ascultă conexiunea și dacă primește o cerere de la un client, va încerca să răspundă trimițând datele către client. Fiecare tip de serviciu are propriile protocoale, care permit clienților să se înțeleagă cu serverele. Pentru FTP, protocolul este FTP sau FTPS (FTP Secure), pentru e-mail protocoalele sunt SMTP (Simple Mail Transfer Protocol), IMAP (Internet Mail Access Protocol) și POP3 (Post Office Protocol ver.3).

În anul 1989, cercetătorul britanic Tim (Timothy) Berners-Lee, care lucra la Organizația Europeană de Cercetări Nucleare (CERN), Geneva - Elveția, a propus un sistem prin care să combine hypertextul[1] cu Internetul, în scopul de a facilita accesul la documentele electronice. A fost pasul care avea să conducă la crearea World Wide Web (WWW sau, simplu, Web).

[1] Hypertext, un text care afișat pe un echipament electronic permite parcurgerea neliniară a documentului, prin intermediul link-urilor.

Tim Berners-Lee

Sursa: https://entrepreneurship.mit.edu/profile/tim-berners-lee/#

Acesta s-a alăturat celorlalte servicii în uz ale Internetului și a devenit, rapid, cel mai utilizat și răspândit dintre toate. Tim Berners-Lee este cel care a creat primul server Web, primul navigator (browser) și protocolul de comunicație între client și server, HTTP (HyperText Transfer Protocol). Totodată a inventat și limbajul cu care se construiesc paginile Web, HTML (HyperText Markup Language - limbaj de marcare a hipertextului).

Îndată după ce a făcut public codul sursă al navigatorului, în 1993, și alți programatori au realizat propriile navigatoare. Între ele, Mosaic și, mai târziu Netscape Navigator, au stat la baza „exploziei" pe care a cunoscut-o Web-ul.

Acest fapt a fost posibil pentru că navigatoarele aveau interfață grafică și puteau fi instalate ușor pe calculatoarele personale ale anilor 90. La aceasta evoluție a contribuit și completa eliminare al limitărilor comerciale în utilizarea Internetului.

Funcționarea Web-ului se bazează pe patru piloni:

1. Clienți - programele care efectuează cereri de pagini web și le afișează utilizatorului. Exemple: Edge, Chrome, Firefox.
2. Servere - programele care primesc cererile clienților, le interpretează și livrează paginile solicitate. Exemple: Apache, IIS, Nginx
3. Protocolul - setul de reguli pe care clientul și serverul le aplică și le respectă pentru a putea face funcțional serviciul (HTTP: HyperText Transfer Protocol). Protocolul are patru faze:
    a. Conectarea - clientul se conectează la serverul care găzduiește pagina. Pentru aceasta folosește serviciul DNS.
    b. Cererea - după conectare, clientul solicită pagina dorită de utilizator
    c. Răspunsul - serverul răspunde cu pagina solicitată sau, dacă aceasta lipsește sau nu este accesibilă, cu un mesaj adecvat.

d. Deconectarea - după servirea răspunsului, serverul întrerupe conexiunea pentru a elibera resurse și a putea primi alte cereri.

4. Limbajul - HTML (HyperText Markup Language). Este limbajul care permite marcajul elementelor dintr-un document pentru ca acestea să poată fi redate conform dorinței autorului/editorului.

Toți acești piloni au fost realizați de Tim Berners Lee, în anul 1990.

Limbajul HTML

Limbaje de marcare

Așa cum sugerează și denumirea, HTML nu este un limbaj de programare ci unul de marcare. Exemple de limbaje de marcare: SGML, XML, MathML. Toate limbajele de marcare se bazează pe etichete (tag-uri) pentru marca porțiuni de text (elemente de text) pentru a realiza o structură de date (XML) sau pentru a stabili modul în care acestea trebuie reprezentate (HTML).

Limbajul XML oferă o mare flexibilitate în definirea etichetelor, acestea putand fi alese liber de editor. Pentru ca un document XML să fie valid, el trebuie să corespundă cu specificațiile unui alt document de tip DTD (Document Type Definition). Acesta stabilește care sunt tag-urile valide și structura documentelor XML. O alternativă mai bună la DTD este XML Schema, care este ea însăși un document XML și permite să fie extinsă prin adăugarea de noi etichete. Conținutul documentelor DTD și XML Schema este stabilite de grupuri de experți din organizațiile care doresc să schimbe date între ele. De fapt, principalul rol al documentelor XML este de a permite schimbul de date între aplicații, fără a avea nevoie de o bază de date. Documentele XML sunt întotdeauna validate pentru a se conforma DTD sau XML Schema, iar cele care nu se conformează nu sunt documente XML valide.

Limbajul HTML este mai restrictiv în privința etichetelor, în sensul că există un set restrâns de etichete ce pot fi folosite, în schimb structura este extrem de flexibilă, existând foarte puține reguli de respectat.

Versiunile limbajului HTML

Prima versiune a limbajului a fost lansată în 1995, sub numele HTML 2.0. La mai puțin de doi ani distanță, în 1997, a fost elaborată versiunea HTML 3.2, prima versiune elaborată de Consorțiul W3C. La sfârșitul aceluiași an au fost elaborate recomandările pentru HTML 4.0 care, după modificări și ajustări, a fost adoptat ca standard ISO/IEC în anul 2000, sub numele HTML 4.01. Acesta a avut și cea mai lungă existență, până în 2014, când au fost publicate, de către W3C, recomandările pentru HTML5. Au urmat, rapid, recomandări pentru HTML5.1 (2016) și HTML5.2 (2017)

După standardizarea versiunii HTML 4.0 a început dezvoltarea unui limbaj separat pe baza specificațiilor XML 1.0. Nu mai este dezvoltat ca un standard separat.

XHTML 1.0 a fost publicat ca o recomandare W3C la 26 ianuarie 2000, și ulterior a fost revizuit și republicat la 1 august 2002. Versiunea XHTML 1.1 a fost publicată în 2001 și se bazează pe XHTML 1.0 Strict. Versiunea XHTML 2.0 a fost abandonată în 2009, în formă de ciornă (draft) deoarece era incompatibilă cu versiunile XHTML1.x iar Consorțiul W3C a dat prioritate dezvoltării specificațiilor pentru HTML5 și XHTML5, acesta din urmă aflându-se tot în stadiu de ciornă.

Toate versiunile limbajelor HTML anterioare HTML5, precum și versiunile XHTML 1.x trebuie să se conformeze unor definiții conținute în documente de tip DTD. Din acest motiv, toate documentele Web încep cu o declarație de tip de document, care comunică navigatorului (clientului Web) ce definiții trebuie să utilizeze pentru redarea corectă a conținutului. Declarațiile folosesc și pentru validarea documentelor. Pentru aceasta există instrumente online care permit validarea diferitelor tipuri de documente, selectarea DTD făcându-se pe baza declarației de tip de document. Declarația de tip de document nu face parte din structura documentului.

Forme de declarații de tip de document

HTML 2.0 - DTD:

```
<!DOCTYPE html PUBLIC "-//IETF//DTD HTML 2.0//EN">
```

HTML 3.2 - DTD:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
```

XHTML Basic 1.0 - DTD:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
"http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
```

Declarațiile de mai sus au mai mult valoare istorică, deoarece în prezent nu se mai utilizează nici una din versiunile limbajelor enunțate.

HTML 4.01

Strict - modul implicit

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

Transitional - permite folosirea de elemente și atribute din versiunile vechi

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
```

Frameset - Pentru documentele care conțin cadre (frames). Acestea sunt descurajate în versiunea 4.01

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
   "http://www.w3.org/TR/html4/frameset.dtd">
```

Detalii despre aceste tipuri de documente, la https://24ways.org/

XHTML1.1

Strict

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

XHTML Basic 1.1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.1//EN"
"http://www.w3.org/TR/xhtml-basic/xhtml-basic11.dtd">
```

HTML5

```
<!DOCTYPE html>
```

HTML5 nu mai este bazat pe DTD și, atunci, declarația de tip se simplifică foarte mult. În plus, dacă la versiunile anterioare declarațiile erau *Case Sensitive* (trebuiau scrise exact ca în exemplele de mai sus, cu minuscule și cu majuscule), la HTML5 se poate scrie oricum, forma `<!doctype html>` fiind perfect valabilă, ca și `<!DOCTYPE HTML>`.

Cu ce editǎm documentele HTML?

Fiind simple texte, documentele HTML pot fi editate, în principiu, cu orice editor de text, inclusiv Notepad. Totuși, pentru ușurarea muncii de editare, există aplicații specializate, cunoscute sub numele de IDE (Integrated Development

Environment), care vin cu o serie de caracteristici de mare ajutor: colorarea sintaxei, sugestii și/sau auto-completări, indentări etc.

Colorarea sintaxei constă în aplicarea de culori diferite diferitelor componente ale documentului, semnalând astfel eventualele greșeli sau omisiuni.

Sugestiile reprezintă funcția de afișare a unei liste de cuvinte cheie pe baza primelor litere tastate de editor. Acesta poate alege din listă cuvântul pe care dorește să îl scrie, fără a trebui să îl tasteze până la capăt. Sugestiile sunt utile și pentru că ajută la evitarea greșelilor de sintaxă.

Autocompletarea este acea funcție care permite, în diverse contexte, completarea automată a unor cuvinte cheie sau structuri, pe baza primelor caractere introduse de editor.

Indentarea reprezintă alinierea textului după anumite reguli, care facilitează citirea și interpretarea codului de către editor.

O listă cu cele mai bune IDE pentru HTML și CSS poate fi găsită aici (Fitzgerald, 2022). De regulă același IDE este folosit atât pentru HTML cât și CSS, deoarece majoritatea oferă suport pentru mai multe limbaje.

Între editoarele enumerate în documentul din link o să prezentăm, pe scurt, trei din cele mai populare, în ordinea complexității.

Indiferent de IDE-ul folosit, documentele editate trebuie salvate într-un director dedicat proiectului, după care pot fi deschise cu oricare navigator (browser) Web.

Ori de câte ori modificați documentul, salvați-l și dați Refresh (F5) în navigatorul cu care vizualizați rezultatul!

Notepad++

Lansarea Notepad++ deschide un document de tip txt. Pentru a activa sintaxa specifică limbajului dorit, din meniul *Language* se alege, imediat după deschidere, limbajul dorit, Figura 3. Apoi, la introducerea textului apare o listă de sugestii din care, cu ajutorul săgeților de la tastatură se alege cuvântul dorit și se apasă tasta Enter sau Tab, Figura 4.

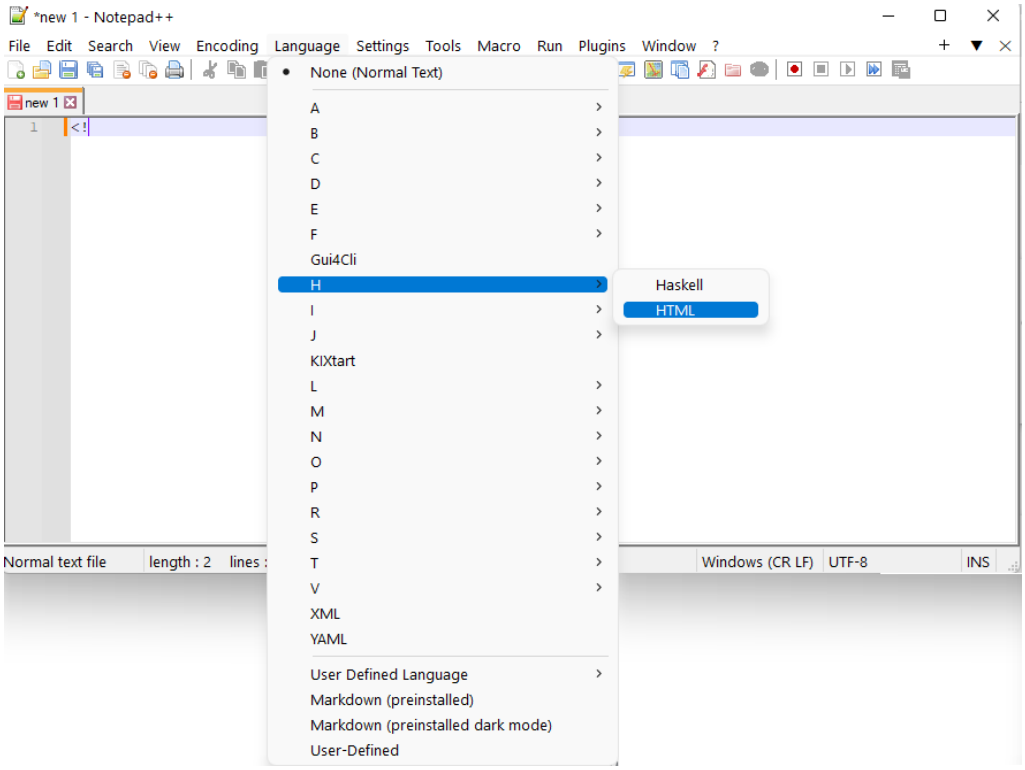


Figura 3. Alegerea limbajului în care se face editarea codului

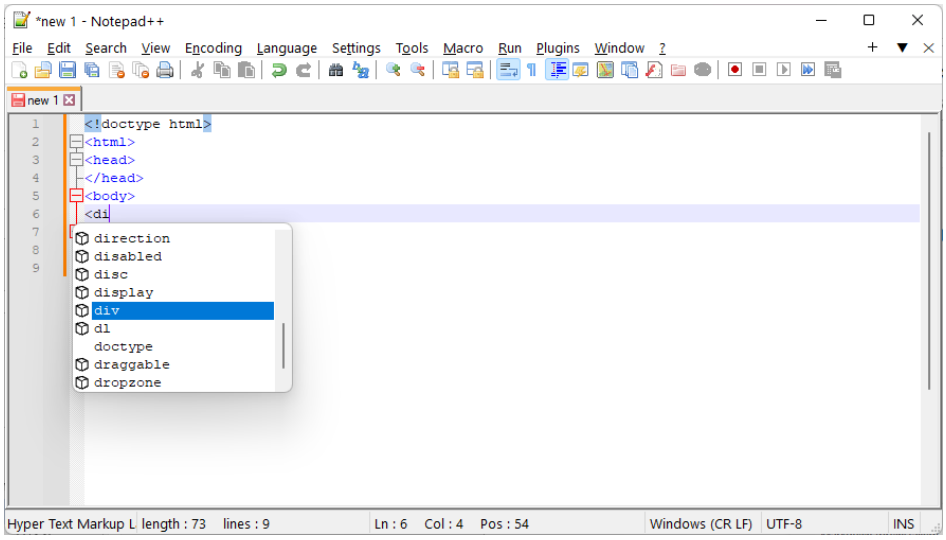


Figura 4. Lista de sugestii

Pentru limbajul HTML este avantajos ca în meniul Settings *->Preferences ->Auto-Completion* să se bifeze opțiunea *HTML/ XML close tag* și opțiunile de completare automată a ghilimelelor duble și simple, *Figura 5*. Asta va face ca tag-urile pereche

să fie completate automat. Pentru limbajul CSS este avantajos să se bifeze și restul opțiunilor de auto-completare a parantezelor rotunde, pătrate și acolade.

Pentru vizionarea rezultatului codului editat se salvează fișierul în folder-ul dorit după care:

1. din meniul *View* alegeți *View Current File in* și apoi browser-ul dorit, Figura 6 SAU
2. deschideți folderul unde ați salvat fișierul, click dreapta pe icon-ul fișierului, apoi alegeți *Open with…*

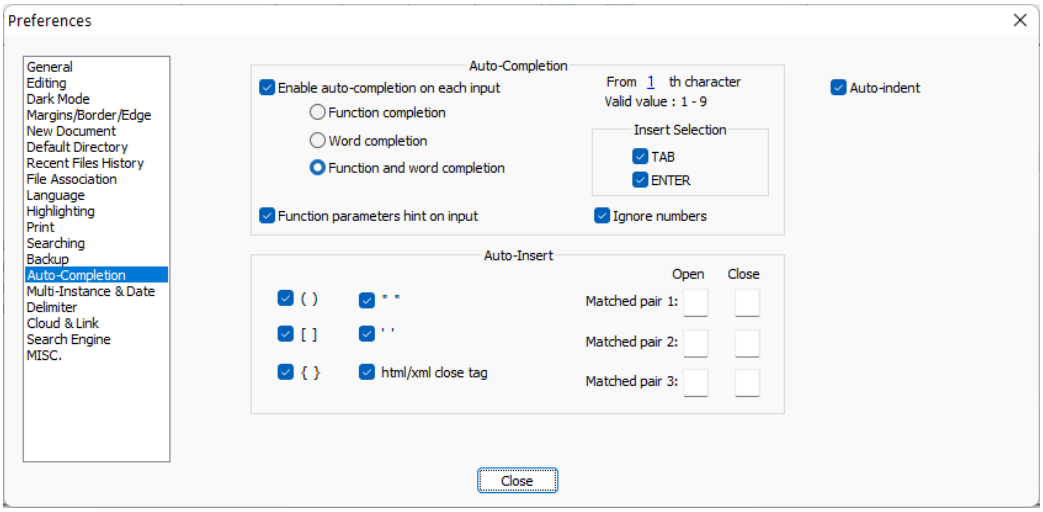


Figura 5. Opțiuni de auto-completare

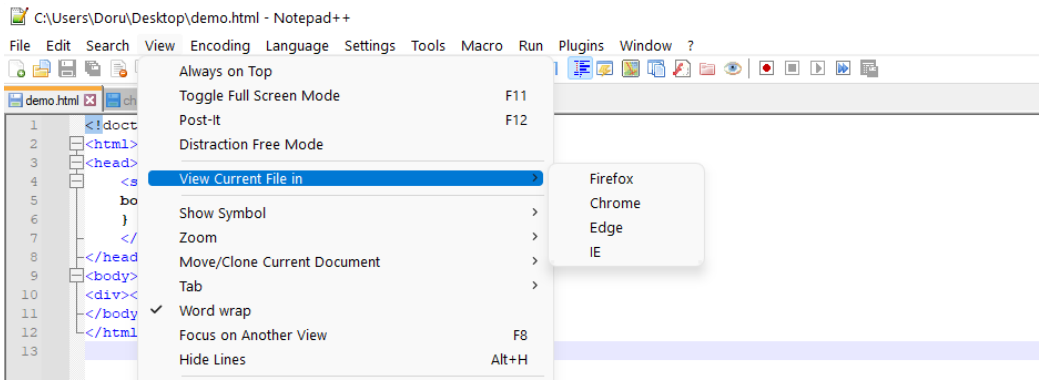


Figura 6. Vizualizarea fișierului curent în browser-ul preferat.

Pentru indentarea conținutului, se apasă tasta TAB, după care toate liniile următoare vor păstra noua indentare. Pentru anularea indentării se apasă tasta Backspace la începutul rândului. Valoarea indentării, în număr de caractere, se stabilește din secțiunea *Language* a meniului *Settings* -> *Preferences*, Figura 7. Valoarea globală, pentru toate limbajele, este de 4 caractere, dar poate fi modificată

individual, pentru fiecare limbaj în parte. O valoare potrivită pentru HTML și CSS este 2, pentru că permite scrierea rândurilor lungi, fără înfășurare (wrap).

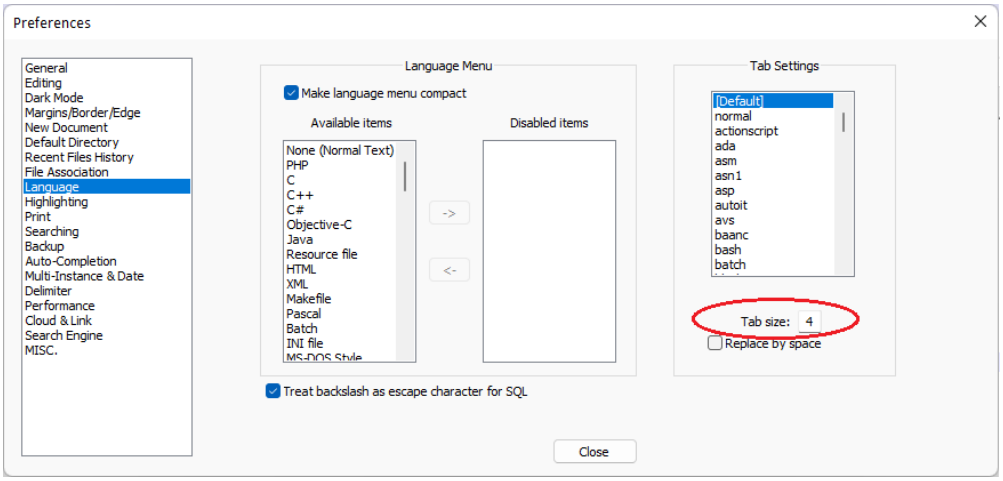


Figura 7. Modificarea valorii tastei TAB (indentarea)

Tema poate fi modificată și personalizată din secțiunea *Settings -> Preferences -> Dark Mode.*

Dimensiunea fontului și tipul de font se pot modifica din meniul *Settings -> Style Configurator*, Figura 8.

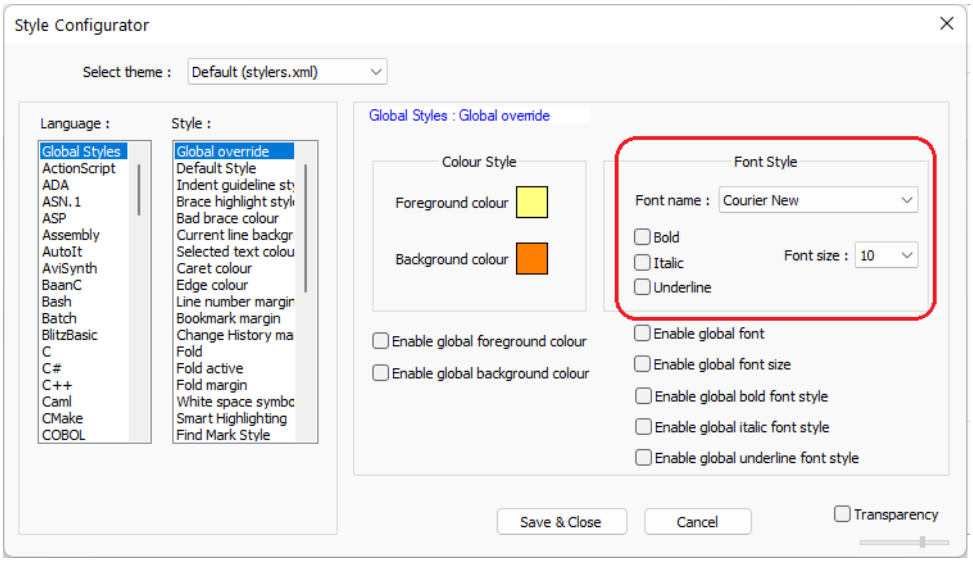


Figura 8. Modificarea fontului

Sublime Text

Sublime Text este un IDE mai avansat decât Notepad++, având funcții de autocompletare mai puternice. Dispune de multe shortcut-uri de la tastatură, ceea ce face editarea mai eficientă. Fereastra poate fi împărțită în 2, 3, sau 4 coloane sau 2 ori 3 rânduri, în fiecare coloană/ rând putând fi editat câte un document Figura 9. Aspectul ferestrei poate fi modificat din meniul *View -> Layout*.

Tot din meniul *View* se poate alege să se afișeze bara laterală (Side bar), care facilitează navigarea printre fișierele deschise.

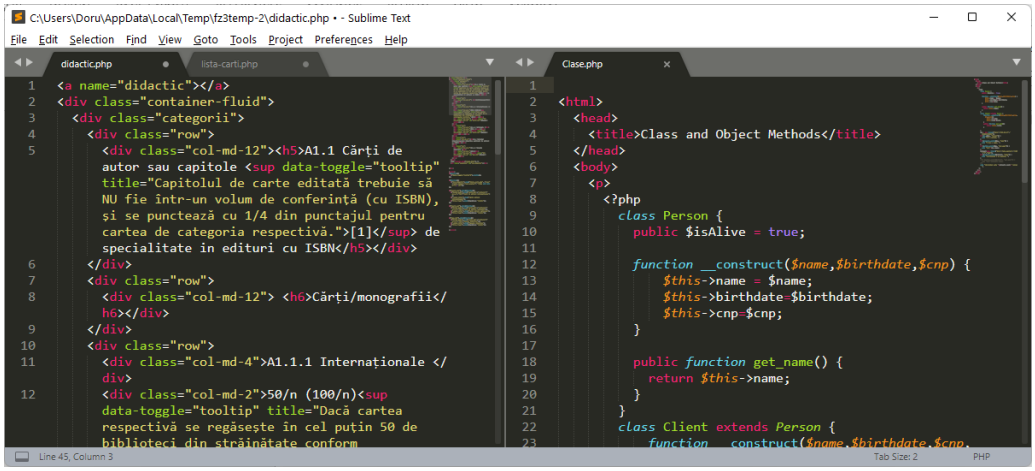


Figura 9. Aspect al IDE cu două coloane

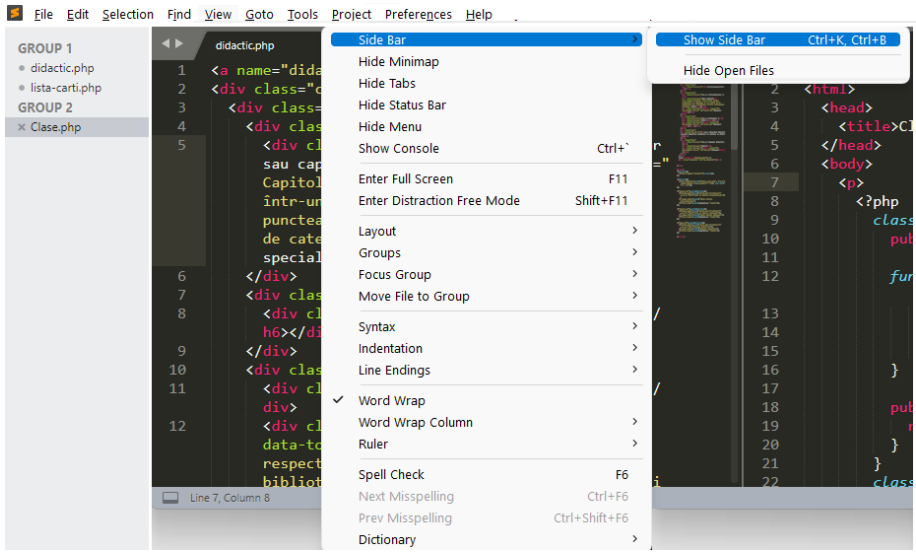
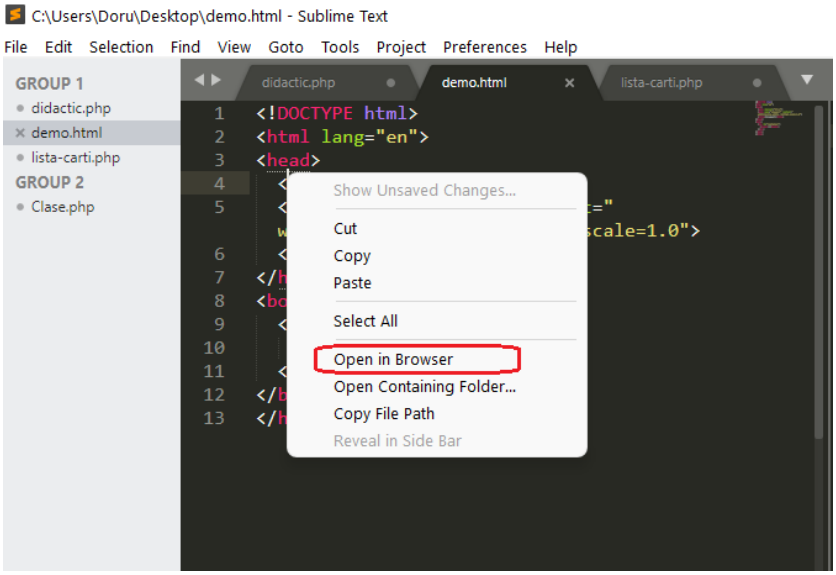


Figura 10 Meniul View. *Afișarea Barei laterale*

Limbajul se alege din meniul *View -> Syntax.* Indentarea se poate regla în intervalul 1-8 din meniul *View -> Indentation.*

Sublime Text dispune de o serie de plugin-uri care pot creste productivitatea. O lista a celor mai importante si utile o găsiți la adresa https://zight.com/blog/sublime-text-plugins/ iar modul de instalare a plugin-urilor la adresa https://johndjameson.com/blog/using-emmet-with-sublime-text.

Pentru vizualizarea paginii, după salvare, se apasă butonul drept al mouse-ului și se alege Open in Browser. Browser-ul în care se deschide pagina este cel implicit al sistemului de operare.



Visual Studio Code (VSC)

Este cel mai avansat IDE și, asemenea Notepad++ este complet gratuit. Complexitatea lui poate fi puțin dezarmantă pentru cei nefamiliarizați iar efortul de învățare a utilizării este semnificativ mai mare decât pentru oricare dintre celelalte. El este mai potrivit pentru scriere programelor in diverse limbaje decât pentru editarea de pagini Web. Figura 11 reproduce fereastra editorului in care 1 este bara de activități, 2 este panoul lateral (Side bar) cu fișierele directorului deschis, 3 este bara de meniuri iar 4 este Panoul de editare (Panoul principal). O parte din panouri, ca și bara de meniuri, pot fi ascunse/afișate din meniul *View -> Appearance*. În Figura 11 panoul de editare este împărțit în trei zone, fiecare având deschisă câte un fișier. Adăugarea/ștergerea de sub-panouri de face din meniul *View -> Editor Layout*, Figura 12

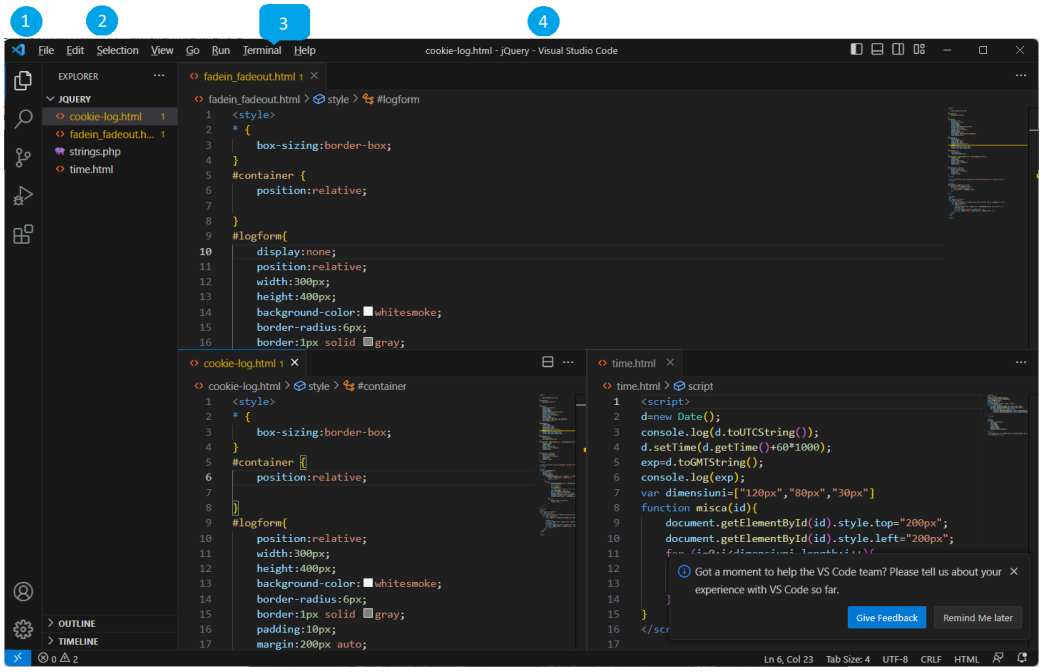


Figura 11. Fereastra IDE Visual Studio Code

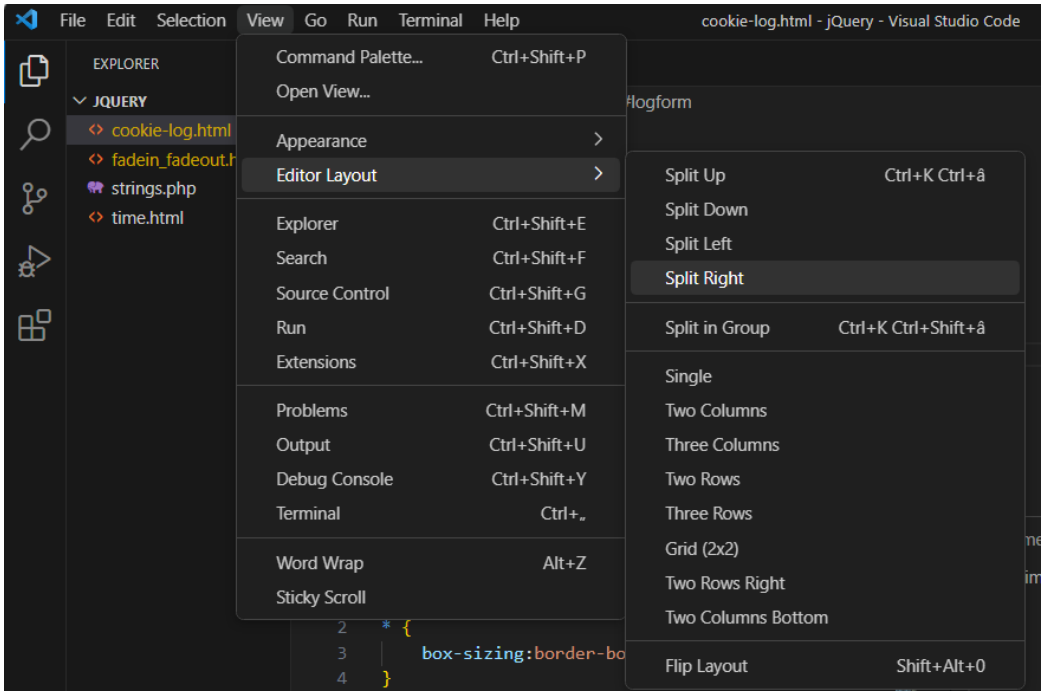


Figura 12. Modificarea aspectului panoului de editare

Modificarea temei se poate face din meniul *File -> Preferences -> Theme* iar tipul de font, mărimea lui, mărimea și controlul indentării se fac din meniul *File -> Preferences -> Settings.*

Pentru a putea previzualiza documentele Web editate, este nevoie de instalarea unei extensii. Există foarte multe extensii pentru VSC, iar pentru previzualizare recomand extensia *HTML Preview*. Despre cum se instalează o extensie găsiți informații la adresa https://www.alphr.com/vs-code-open-in-browser/. Odată instalată extensia, în panoul de editare al documentului dați click dreapta pe mouse și alegeți *Show in Browser*.

Structura documentelor HTML

Un document HTML este un text care conține marcaje sub forma de tag-uri (etichete). Unele tag-uri folosesc pentru a semnala modul de redare a fragmentelor de conținut, altele la structurarea documentului. Tag-urile sunt de două tipuri: pereche și nepereche sau auto-închise. Toate tag-urile au un nume, iar în cazul etichetelor pereche acestea sunt sub forma `<tagname>.....</tagname>`, adică eticheta de început conține numele tag-ului încadrat între simbolurile `<` și `>` iar eticheta de sfârșit este similară celei de început, dar numele este precedat de simbolul slash `/`.

Tag-urile nepereche nu au etichetă de sfârșit, fiind de forma: `<tagname>` .

HTML5 permite scrierea etichetelor atât cu majuscule cât și minuscule sau combinații, însă recomandarea este să se folosească minuscule.

Unele tag-uri atât pereche cât și nepereche acceptă atribute, în forma:

```
<tagname atribut1="valoare" atribut2="valoare">
```

Rolul atributelor este de a furniza informații despre elementul marcat de `tagname`.

Pentru structurarea documentului se folosesc tag-urile pereche `<html>`, `<head>` și `<body>`.

```
<html>
<head>
..Conținut antet. Acesta nu este vizibil în fereastra navigatorului.
</head>
<body>
..Conținut document. Acesta este vizibil în fereastra navigatorului.
```

```
</body>
</html>
```

Întregul document este încadrat între tag-urile `<html>` și `</html>`. Acestea marchează începutul și sfârșitul documentului. După eticheta `</html>` nu poate urma nimic, iar în fața tag-ului `<html>` doar declarația de tip de document trebuie să apară. Eticheta de început poate conține un atribut, *lang*, care specifică limba în care este redactat documentul. Este util motoarelor de căutare, când se aplică filtre pentru rezultatele căutării.

Exemple: `<html lang="fr">`, `<html lang="en">`, `<html lang="ro">`.

Tag-uri pentru antet

Antetul, cuprins între tag-urile `<head>` și `</head>` conțin meta informații (meta data), adică informații despre document. Aceste meta informații sunt marcate, de asemenea, cu etichete, atât pereche cât și nepereche. Mai jos dăm o listă cu principalele etichete meta:

```
<meta charset="nume_set_caractere">
```

stabilește setul de caractere care trebuie folosit de navigator la afișarea caracterelor care nu aparțin setului de caractere Latin-1 (caracterele alfabetului latin), cum ar fi diacriticele pentru limba română, ideogramele limbilor asiatice, alfabetul ebraic, armean etc.).

Setul de caractere *utf-8* este cel mai cuprinzător, codificând pe mai mulți octeți (cod multi-octet) toate caracterele din limbile cunoscute, simboluri extrem de variate precum ♫ ¥ ☺ ⅝ ☯ 亜  ↭ ℜ  etc. și dispune de foarte multe coduri încă neatribuite. O lectură scurtă, interesantă și accesibilă ca terminologie găsiți la https://www.w3.org/International/.

Prin urmare, paginile scrise în alte limbi decât latină și engleză au nevoie de eticheta

`<meta charset="utf-8">`[2]. Aici se impune o observație: este **esențial** ca editorul cu care se scrie codul HTML să folosească aceeași codificare ca cea setată în eticheta meta charset. Cu alte cuvinte, codificarea caracterelor speciale să se

[2] Și această etichetă s-a simplificat în HTML5. La versiunile anterioare, ea arăta astfel:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

facă identic la editare ca și la afișare. Despre acest aspect vom discuta într-o secțiune separată.

Alte etichete meta importante, în special pentru motoarele de căutare, sunt *description* și *keywords*:

```
<meta name="description" content="descrierea succintă și elocventă a conținutului paginii">
<meta name="keywords" content="listă, 'de cuvinte', cheie, conținute, 'în pagină'">
```

După cum rezultă din exemplele de mai sus, atributul *content* al tag-ului *description* conține o descriere sumară, dar cât mai fidelă a ceea ce se găsește în pagină. Numărul recomandat de caractere este între 50 și 160. Google trunchiază descrierile de peste 160 de caractere, deci tot ce e în plus nu e valorificat de motorul de căutare.

Spre exemplu, inspectarea paginii USV evidențiază prezența tag-ului *meta description* cu următorul conținut:

```
<meta name="description" content="USV este una dintre cele mai dinamice universități din Europa de Est oferind o educație universitară și post-universitară atractivă și de calitate. USV asigură un curriculum cuprinzător, cu mai mult de 100 de programe de studii la licență, masterat, doctorat și nivel post-doctoral pe diverse domenii: socio-umane tehnice, economice, științe ale naturii."> == $0
```

Căutarea cu Google după cuvintele „Universitatea Suceava” afișează printre rezultate aceste informații:

Figura 13 Conținutul tag-ului description apare ca descriere a link-ului către pagina USV

Se constată cu ușurință că descrierea de sub link-ul rezultat este preluată din eticheta meta *description*. Mai multe informații utile aici: https://moz.com/learn/seo/meta-description.

A doua etichetă, keywords, ajută la o mai bună indexare a motoarelor de căutare, dar nu toate. De exemplu, Google ignoră informațiile conținute de atributul *content*. Cuvintele cheie trebuie să fie din cele aflate în pagină, dintre cele mai

reprezentative pentru conținutul ei. Cuvintele compuse se scriu între ***ghilimele simple***.

Alte etichete meta:

robots - conține directive pentru motoarele de căutare

`<meta name="robots" content="index, follow">` spune roboților să indexeze pagina și paginile care au link în pagina curentă. Alte valori posibile sunt `noindex, nofollow` și orice combinație între ele, de exemplu `index, nofollow.`

author - specifică numele autorului/ autorilor paginii

```
<meta name="author" content="Paul Mocanu, Betty Pavel">
```

viewport[3] - setează viewport-ul pentru ca pagina să arate bine pe toate dispozitivele:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Mai multe despre tag-urile meta, în capitolul rezervat SEO (Search Engine Optimization)

O etichetă foarte importantă, este eticheta `<title>`. Între eticheta de început și cea de sfârșit se scrie titlul paginii, care este foarte important pentru motoarele de căutare. Cu cât titlul este mai inspirat și rezumă mai bine conținutul paginii, cu atât pagina va obține o poziționare mai bună în rezultatele căutării. Totodată, titlul paginii apare pe tab-ul paginii, în navigator. Spre exemplu, titlul paginii USV este cel de mai jos:

```
<title>
  "Universitatea „Ștefan cel Mare” din Suceava - Una dintre cele mai dinamice
  universități din Europa de Est" == $0
</title>
```

iar rezultatul este cel din Figura 14. Dacă analizăm Figura 14, constatăm că textul link-ului din pagina cu rezultatele căutării este identic cu cel al filei (tab-ului) paginii.

[3] viewport-ul este zona vizibilă a utilizatorului pe o pagină web. Acesta variază în funcție de dispozitiv - mai mic pe un telefon mobil decât pe ecranul unui computer.

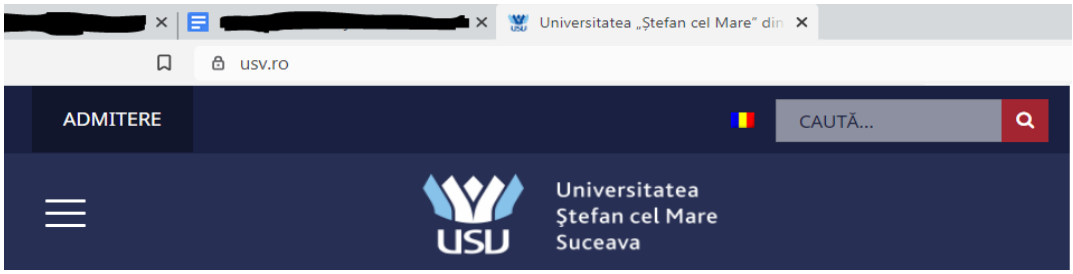


Figura 14. Titlul paginii apare pe tab-ul paginii, în navigator

În rezumat, antetul unei pagini ar trebui să conțină minim elementele de mai jos:

```
<head>
  <meta charset="utf-8">
  <title>Titlul "SMART" al paginii</title>
  <meta name="description" content="Descrierea fidelă a paginii, între 50 și 160 de caractere">
  <meta name="keywords" content="lista de cuvinte cheie, separate cu virgulă">
  <meta name="robots" content="index, follow">
  <meta name="author" content="numele autorilor">
</head>
```

Tot în antet se mai introduc etichete care fac legătura cu fișiere externe care conțin scripturi (programe care rulează în navigator) sau definiții de stil pentru elementele paginii. Despre acestea din urmă vom discuta la capitolul despre limbajul CSS.

Tag-uri pentru BODY

O parte din tag-urile HTML5 sunt moștenite de la HTML 4 iar dintre acestea, unele au fost eliberate de o serie de atribute, informațiile despre detaliile afișării fiind preluate de CSS.

Și tag-urile pentru BODY sunt pereche și nepereche (auto închise). De regulă, tag-urile pereche au comportament de *bloc*, adică se comportă de parcă ar fi precedate de tag-ul
 și urmate tot de o tag-ul
. Elementelor de tip *bloc* li se poate stabili o lățime și o înălțime. Există și excepții: tag-ul SPAN este pereche și nu are comportament de *bloc*. Similar se comportă CITE, DEL, INS, ABBR și altele.

Tag-urile auto închise au, de regulă, comportamente de elemente *inline* (în linie) cu elementele care le preced și le urmează. Cele două cazuri sunt ilustrate în Figura 15:

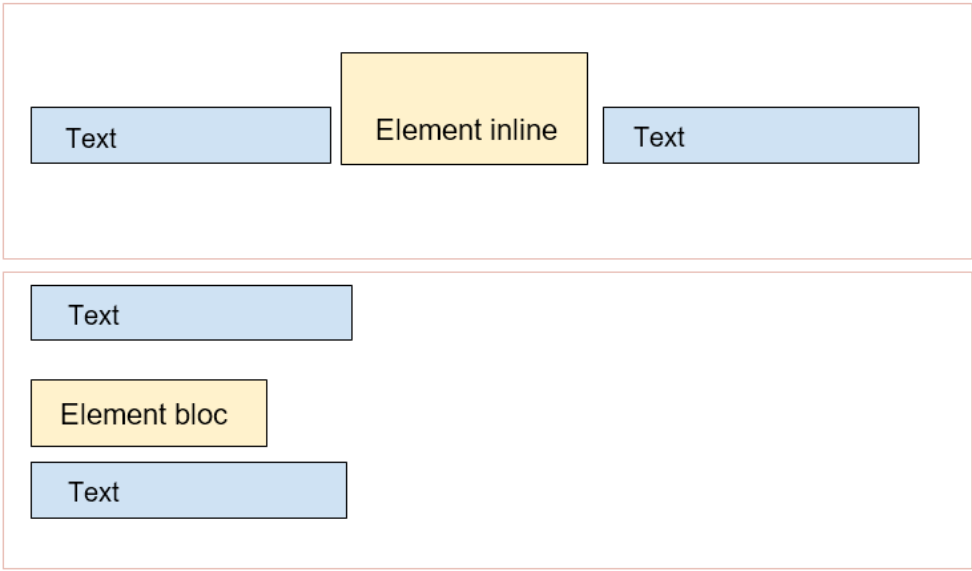


Figura 15. Tipuri de comportamente ale elementelor HTML

Există și aici excepții, de exemplu tag-ul IMG, nepereche, are comportament de tip *inline-block*, adică este *inline* dar acceptă atribute pentru dimensiuni.

Ambele tipuri de tag-uri acceptă atribute. Există un set de atribute globale, pe care le acceptă toate tag-urile, dar unele tag-uri au atribute specifice.

Tabel 1. Atribute HTML globale

Nume atribut	Descriere
accesskey	Stabilește o scurtătură (shortcut) pentru focus pe un element
class	Specifică una sau mai multe clase pentru un element
contenteditable	Specifică dacă un element este sau nu editabil. Valori: „true” | „false”
draggable	Specifică dacă un element este „târâbil” sau nu. Valori: „true” | „false” | „auto”. Necesită cod JavaScript
hidden	Specifică dacă elementul este ascuns sau nu.
id	Specifică un identificator unic pentru un element
lang	Specifică limba în care este scris conținutul elementului
style	Specifică atributele de stil ale elementului, în linie cu tag-ul
tabindex	Specifică ordinea elementului la apăsarea tastei Tab

title	Specifică extra informații pentru un element

Exemple de utilizare a c\torva dintre ele la adresa https://playcode.io/707694/

Tag-uri moștenite de la HTML 4

Tag-uri pentru tipuri de scriere

Nume	Tip	Comportament	Descriere
H1..H6	pereche	bloc	titlu
P	pereche	bloc	paragraf
BR	nepereche	bloc?	salt la linie nouă
SPAN	pereche	inline	container inline pentru stilizarea unei părți din document
PRE	pereche	bloc	text preformatat
HR	nepereche	bloc	riglă orizontală (mai puțin folosită, se preferă CSS)
A	pereche	inline	ancora (pentru marcaje în document sau link-uri)
B	pereche	inline	stil bold
STRONG	pereche	inline	evidențiere prin îngroșare (bold)
I	pereche	inline	stil italic
EM	pereche	inline	evidențiază prin înclinare (italic)
BIG	pereche	inline	stil de scris mai mare
SMALL	pereche	inline	stil de scris mai mic
SUB	pereche	inline	subscript
SUP	pereche	inline	superscript
CITE	pereche	inline	citare. Spre deosebire de BLOKQUOTE, este *inline*

BLOCKQUOTE	pereche	bloc	citare bloc de text lung
CODE	pereche	inline	computer code fragment
SAMP	pereche	inline	exemplu de cod calculator. Similar cu CODE
OL	pereche	bloc	listă numerotată (ordonată)
LI	pereche	bloc	articol de listă (list item)
UL	pereche	bloc	lista marcată (ne numerotată)
DL	pereche	bloc	Definește o listă
DT	pereche	bloc	Definește un termen al listei
DD	pereche	bloc	Descrie o definiție Vezi exemplu de utilizare
DEL	pereche	inline	marchează textul ca șters
INS	pereche	inline	text inserat
ABBR	pereche	inline	abreviere. Se folosește împreună cu atributul TITLE
ADDRESS	pereche	bloc	Informații despre autor

Spre exemplu, următorul cod HTML:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta    name="viewport"    content="width=device-width,
initial-scale=1.0">
    <link rel="stylesheet" href="src/style.css">
  </head>
  <body>
    <h1>Acesta este un titlu H1</h1>
    <h2>Acesta este un titlu H2</h2>
    <p>Un paragraf oarecare</p>
    <p>Urmat de un al doilea paragraf</p>
    <div>Acesta este un DIV</div>
    <div>Urmat de un al <span>doilea</span> DIV</div>
```

```
    <cite>Un citat CITE este scris oblic.</cite>
    <code>O secventa cu CODE</code>
  </body>
</html>
```

produce rezultatul redat în Figura 16.

De observat că blocurile de tip paragraf (P) prezintă o distanță deasupra și dedesubt, față de alte elemente din document, în timp ce blocurile de tip DIV nu au această distanțare. Deși liniile CITE și CODE sunt scrise în sursa unele sub altele, în rezultat ele apar una după alta, deoarece tag-urile sunt de tip inline.

Mai observăm că tag-ul SPAN nu modifică în nici un fel modul de afișare a conținutului celui de al doilea bloc DIV. De fapt, limbajul HTML ignoră extra-spațiile (mai mult de un spațiu) și salturile la linie nouă din sursa paginii.

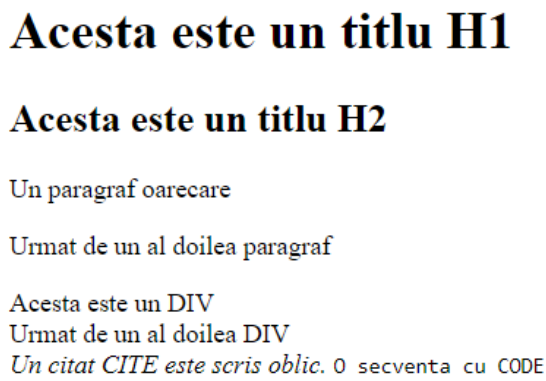


Figura 16. Rezultatul codului din exemplul anterior

Codul următor produce fix același rezultat ca cel anterior:

```
<h1>Acesta este un titlu H1</h1> <h2>Acesta este un titlu H2</h2>
     <p>Un paragraf oarecare</p> <p>Urmat de un al doilea paragraf</p>
     <div>Acesta este un DIV</div><div>Urmat  de un al <span> doilea</span> DIV</div>
```

Aranjarea codului prin alinieri (inventări) și așezarea liniilor unele sub altele este utilă celui care proiectează pagina, fiindu-i mai ușor să identifice elementele de conținut din pagină. Cu siguranță codul de mai jos este mai inteligibil decât cel din Figura 17.

1. Exemplu de utilizare tag-uri prezentate
2. http://cdn.mos.musicradar.com/audio/samples/90s-ambient-demos/MiniKit_01(Full).mp3

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="src/style.css">
  </head>
  <body>
    <h1>Acesta este un titlu H1</h1>
    <h2>Acesta este un titlu H2</h2>
    <p>Un paragraf oarecare</p>
    <p>Urmat de un al doilea paragraf</p>
    <div>Acesta este un DIV</div>
    <div>Urmat de un al <span>doilea</span> DIV</div>
    <cite>Un citat CITE este scris oblic.</cite>
    <code>O secventa cu CODE</code>
  </body>
</html>
```

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="src/style.css">
  </head>
  <body><h1>Acesta este un titlu H1</h1><h2>Acesta este un titlu H2</h2><p>Un paragraf oarecare</p>
    <p>Urmat de un al doilea paragraf</p><div>Acesta este un DIV</div>
    <div>Urmat de un al <span>doilea</span> DIV</div><cite>Un citat CITE este scris oblic.</cite>
    <code>O secventa cu CODE</code></body></html>
```

Figura 17.Exemplu de indentare a codului. Codul din imaginea de sus este mai ușor de citit de către editorul paginii decât cel de jos.

Alte tag-uri

Nume	Tip	comportament	Descriere
IMG	nepereche	inline-block	definește o imagine având ca sursă fișierul specificat de atributul SRC

TABLE	pereche	bloc	definește un tabel
CAPTION	pereche	bloc	adaugă o legendă tabelului
TR	pereche	bloc	definește un rând
TH	pereche	inline - bloc	definește o celulă header
TD	pereche	inline - bloc	definește o celulă obișnuită

Tag-urile TABLE, TH și TD admit atributul *width* cu valori numerice sau procentuale. Valorile numerice nu au unitate de măsură, dar reprezintă numărul de pixeli:

```
<table width="500">
…
</table>
```

```
<table width="60%">
…
</table>
```

Un exemplu cu tabel simplu puteți vedea aici, iar unul mai complex, aici. Acest ultim exemplul conține și o imagine pe ultimul rând.

Din păcate, fără CSS nu se poate obține mai mult ca aspect.

Elementul IMG. Permite inserarea unei imagini într-un document. Este un element inline și cere minim atributul *src*, care specifică sursa imaginii:

```
<img src="cale/nume_fisier">
<img src="URL">
```

În prima formă, fișierul se află pe același calculator cu documentul, la pe o altă cale. În al doilea exemplu sursa fișierului imagine este specificată printr-o adresă Web (URL).

Tipurile de fișiere imagine cele mai utilizate sunt jpg, jpeg, gif, png, svg și webp. Documentația completă a tipurilor de fișiere acceptate este disponibilă pe site-ul developer.mozilla.org.

Pentru a înțelege ce este calea, vom considera următoarea structură de fișiere, Figura 18:

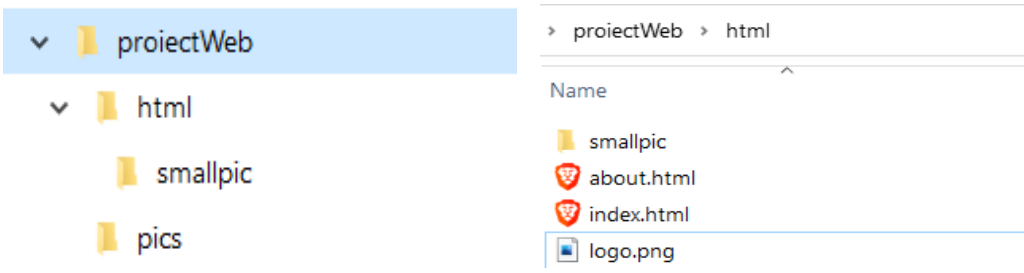


Figura 18. Structură de fișiere pentru exemplificarea scrierii căii

Întregul proiect este conținut de folderul (directorul) *proiectWeb*. Acesta conține două directoare, *html* și *pics*, în ideea în care primul va conține doar fișiere de tip html iar al doilea doar imagini. Directorul *html* conține și el un subdirector, *smallpics*, unde sunt stocate mici iconițe care apar în pagini și un fișier imagine *logo.png*, care conține logo-ul site-ului.

Dacă logo-ul trebuie inserat în pagina index.html, atunci se va scrie:

```
<img src="logo.png">
```

Dacă în aceeași pagină trebuie inserată imaginea *arrow.png* din directorul *smallpic*, atunci se va scrie:

```
<img src="smallpic/arrow.png">
```

care se citește astfel: din locul unde ne aflăm (directorul html), intră în directorul *smallpic* și ia fișierul arrow.png. Calea este *smallpic/* . Simbolul slash / este un separator pentru elementele căii. Dacă în aceeași pagină trebuie inserată imaginea *landscape.jpg* directorul *pics*, atunci se va scrie:

```
<img src="../pics/landscape.jpg">
```

care se citește: ieși din directorul curent (../), intră în directorul *pics* și alege *landscape.jpg*.

Există o reprezentare specială a unor directoare: ./ înseamnă directorul curent iar ../ înseamnă directorul cu un nivel deasupra. Din această perspectivă, inserarea imaginii săgeții *arrow.png* se poate scrie și `<img src="./smallpic/arrow.png">`

Acest mod de referire la o (re)sursă se numește „relativ” pentru că referirea se face relativ la poziția curentă a fișierului care apelează resursa. Astfel, dacă mutați directorul *proiectWeb* pe un suport extern (memory stick), pe un alt calculator sau pe un CD, paginile vor funcționa fără probleme. Dacă există o referire către o resursă din afara directorului *proiectWeb*, la mutarea pe un alt mediu de stocare, acea resursă va deveni indisponibilă.

Mai există o metodă de referire, ***absolută***, care precizează calea absolută către fișierul sursă, de exemplu:
C:/Users/Current_user/proiectWeb/html/smallpic/arrow.png. Acest mod de referire trebuie evitat cu orice preț, deoarece dacă directorul *proiectWeb* se mută pe un alt mediu și va fi testat pe alt calculator (gazdă), cel mai probabil nu va funcționa pentru că pe noua gazdă calea respectivă să nu existe (spre exemplu diferă *Current_user*)

Deși este afișat inline, elementul IMG admite atributele *width* și *height* care stabilesc lățimea și respectiv înălțimea imaginii. În lipsa lor, imaginea este afișată cu dimensiunile originale. Se exemplu, dacă imaginea are 600x400px (W x H) și dorim să o afișăm cu lățimea de 300 px, vom scrie `<img src="path/fisier" width="300" height="200">` sau mai bine `<img src="path/fisier" width="300">`, pentru că astfel atributul height capătă valoarea AUTO și stabilește automat valoarea înălțimii astfel încât raportul W/H să se păstreze. Altfel, va trebui să calculăm de fiecare dată valoarea înălțimii

$$Hnou = Hvechi * \frac{Wnou}{Wvechi}$$

Desigur că se poate stabili valoarea celeilalte dimensiuni, H, rămânând ca lățimea să fie stabilită automat `<img src="path/fisier" height="200">`

În general nu se recomandă redimensionarea imaginii în navigator pentru că o imagine de dimensiuni mari se încarcă greu iar redimensionarea consumă resurse ale calculatorului și experiența vizitatorului pe pagină nu va fi una plăcută. În niciun caz nu vom reduce o imagine de 1000 x 750 px la 200 x 150 px. Mici ajustări, de 20..30% pot fi acceptabile.

Pentru situația în care avem nevoie de două versiuni ale aceleași imagini, vom avea două surse diferite, cu dimensiunile potrivite, de exemplu “landscape_small.jpg” și “landscape.jpg”. Afișam miniatura sub formă de link iar cine dorește să vadă imaginea la dimensiunile reale, va da click pe miniatură.

```
<a href="imagini/landscape.jpg" target"_blank">
  <img src="imagini/landscape_small.jpg" height="200">
</a>
```

Stabilirea dimensiunilor este importantă în faza de definire a aspectului paginii (layout), când imaginile nu sunt disponibile dar în machetă trebuie să li se rezervă spațiul necesar.

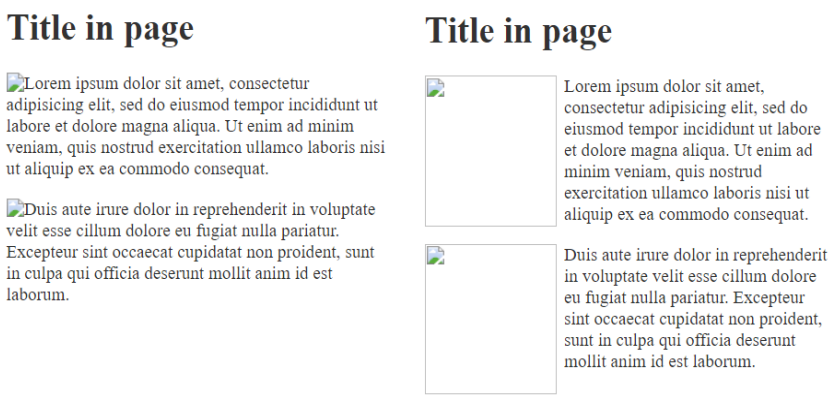


Figura 19. Utilizarea atributelor HEIGHT și WIDTH pentru rezervarea locului imaginii în aspectul paginii

Figura 19 ilustrează situațiile în care dimensiunile unei imagini care nu este disponibilă nu sunt stabilite (stânga) și când sunt stabilite (dreapta).

Un alt atribut recomandat al tag-ului IMG este ALT, care conține o descriere a conținutului imaginii. El este util motoarelor de căutare dar și persoanelor cu deficiențe de vedere. Utilizarea atributului ALT anulează efectul atributelor WIDTH și HEIGHT pentru imaginile ce nu pot fi afișate (fișier lipsă, cale specificată greșit).

Elemente Multimedia

Tipurile de elemente multimedia folosite sunt, filmele, înregistrările audio, documente pdf.

Imagini și sunet

Fișierele video cele mai utilizate ca sursă pentru elementele multimedia sunt mp4 (recomandat pentru că este larg acceptat), webm (Web Media File, de asemenea larg acceptat, compatibil cu mp4), avi Audio Video Interleave (preferat pentru clipuri scurte, având dimensiune mare), wmv (Windows Media Video) și ogg (Jacklin, 2023)

Pentru surse audio, cele mai folosite sunt tipurile mp3, wav (Waveform Audio File Format) și ogg.

În codul de mai jos, disponibil la adresa https://playcode.io/1496889 avem două elemente, unul audio și altul video, ambele cu atributul *autoplay*. Ambele elemente folosesc surse externe (de pe alte servere).

```
<!DOCTYPE html>
<html>
<head>
     <title>Audio-video</title>
</head>
<body>
<audio controls autoplay>
     <source            src="https://sampleswap.org/samples-
ghost/MELODIC%20LOOPS/GUITAR%20LOOPS/459[kb]090_tight-wakka-
tone-guitar-groove-1.wav.mp3" type="audio/mpeg">
     <p>Browser-ul   tau   nu   suporta   elementul   HTML5
<i>audio</i>.</p>
     </audio> <p></p>
<video controls width="400" autoplay>
     <!--                                                  <source
src="https://archive.org/download/ElephantsDream/ed_1024_512
kb.mp4" type="video/mp4"> -->
     <source
src="https://cdn.bitdegree.org/learn/Pexels%20Videos%203373.
mp4" type="video/mp4">
     <p>Browser-ul   tau   nu   suporta   elementul   HTML5
<i>video</i>.</p> -->
</video>
</body>
</html>
```

1. Copiați codul și creați un document nou ‘audio-video.html’ și lipiți codul în document.
2. Comentați sursa audio și introduceți o sursă nouă, de pe Internet. Salvați!
3. Deschideți documentul în Chrome sau Firefox și verificați funcționarea.
4. Deschideți documentul în Edge și verificați funcționarea atributului *autoplay*.
5. Decomentați prima sursa video și comentati-o pe a doua. Salvați!
6. Dați Refresh paginii, în ambele navigatoare: Chrome/Firefox și Edge.
7. Adăugați atributul *loop* la ambele elemente și verificați rezultatul.
8. Ștergeți atributul width=”400” al etichetei VIDEO și notați rezultatul.
9. Adăugați atributul height=”250” și notați rezultatul.
10. Adăugați, pe lângă atributul height și atributul width=”600” și notați rezultatul.

11. Ce concluzii trageți de la punctele 8, 9 și 10?

Acum accesați adresa https://ie.usv.ro/media/media_error.html, unde pentru simplitate am lăsat o singură sursă audio și una video.

Veți constata că fișierul video nu se încarcă. Dacă apăsați tasta F12 și selectați 'Console' ca în figura 1, veți constata un text scris cu roșu (Figura 20):

Failed to load resource:

net::ERR_CONNECTION_REFUSED

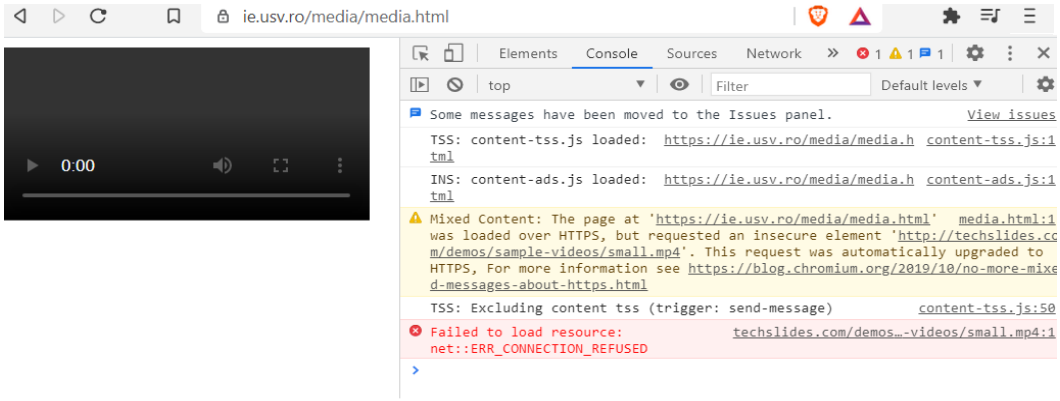


Figura 20. Eroare de refuz de conexiune

Această eroare apare ori de câte ori se încearcă încărcarea unei resurse de pe un server cu conexiune nesecurizată într-un document HTML de pe un server cu conexiune securizată. Serverul ie.usv.ro folosește o conexiune securizata cu protocol HTTPS (HTTP Secure) în timp ce resursa se află pe un server cu conexiune nesecurizată (http://techslides.com)

Pe calculatorul vostru va funcționa fără probleme, deoarece protocolul *file:///*, folosit de navigator pentru a deschide fișiere locale, nu este unul securizat.

Resursele afectate de această limitare sunt imaginile, fișierele audio, video, fișierele cu definițiile de stil CSS și fișierele cu programe (scripturi) JavaScript.

Pentru mai multe detalii despre cum să atașați o imagine care să fie afișată în locul fișierului video, înainte ca acesta să fie redat, despre raportul de aspect, despre preîncărcare fișierelor media și despre cum să pornească playerul cu sunetul oprit inițial, accesați:

https://developer.mozilla.org/en-US/docs/Learn/HTML/Multimedia_and_embedding/Video_and_audio_content.

Încercați și varianta:

```
<audio controls src="https://sampleswap.org/samples-ghost/MELODIC%20LOOPS/GUITAR%20LOOPS/459[kb]090_tight-wakka-tone-guitar-groove-1.wav.mp3">

<video controls width="400" src="https://cdn.bitdegree.org/learn/Pexels%20Videos%203373.mp4">
```

Care ar fi avantajele și dezavantajele acestei variante față de varianta cu specificarea sursei prin elementul SOURCE?

Inserare clipuri YouTube

YouTube oferă posibilitatea de a încorpora clipurile video găzduite pe platformă în paginile web ale utilizatorilor. Pentru aceasta urmați pașii de mai jos:

1. Pe un computer, accesați videoclipul sau lista de redare YouTube pe care doriți să-l încorporați.
2. Faceți clic pe Trimite (Share) Figura 21.
3. Din lista de opțiuni de Partajare, faceți clic pe Încorporează (Embed), Figura 22.
4. Din caseta care apare, copiați codul HTML, Figura 23
5. Lipiți codul în HTML site-ul dvs.

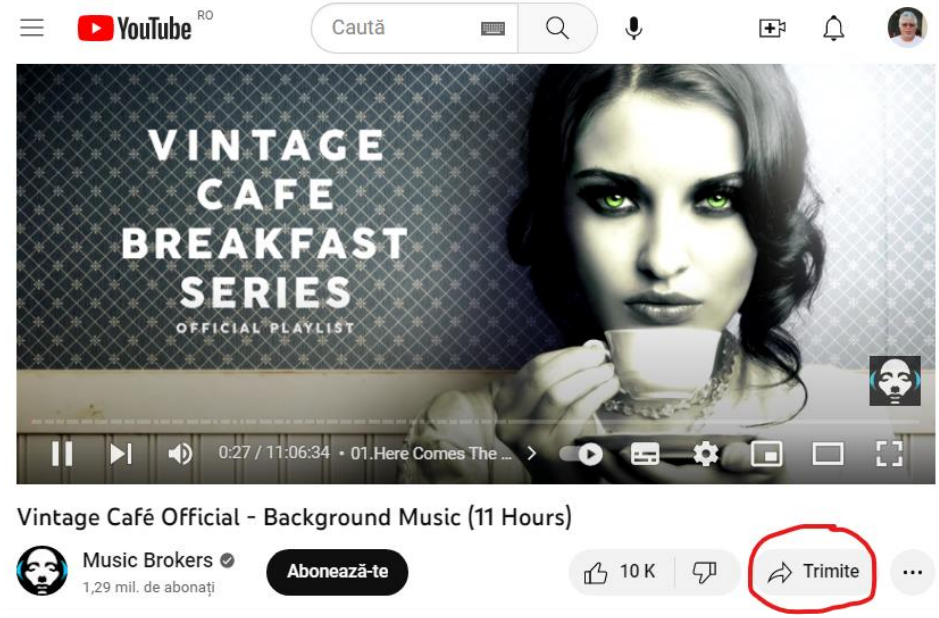


Figura 21. Se caută și se apasă butonul Trimite (Share)

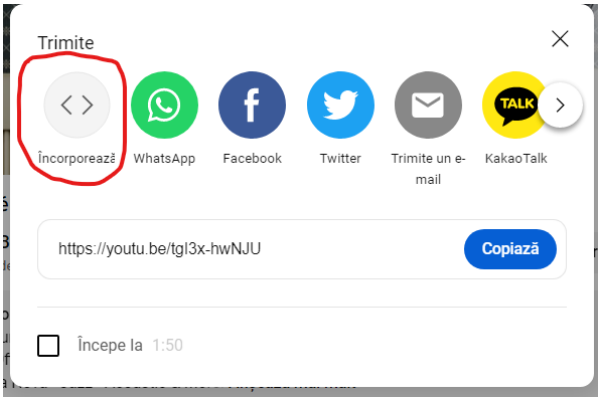


Figura 22. Se alege Încorporează (embed)

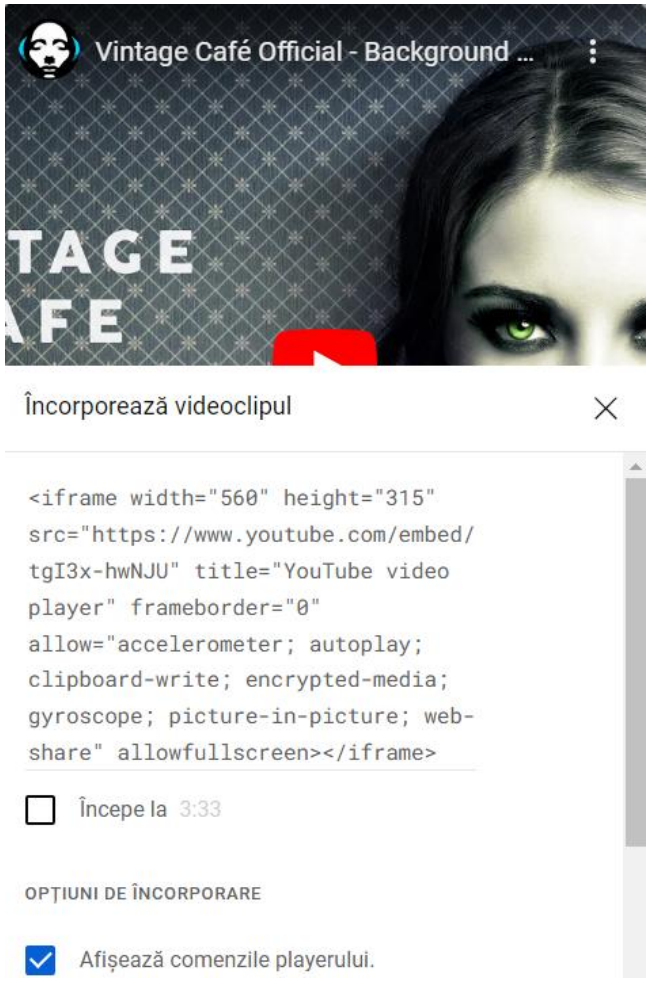


Figura 23. Se copie codul generat de YouTube și se inserează în poziția dorită

Inserare documente pdf

De regulă, paginile Web oferă link-uri către documentele pdf, pentru că acestea se deschid într-un cititor specializat (ex: Adobe Acrobat sau o altă filă a navigatorului) care permit o lecturare mai ușoară.

```
<a href="cale/fisier.pdf">Click pentru a descărca fișierul</a>
```

Dacă se dorește inserarea unui document pdf chiar în interiorul paginii Web, se folosește unul din tag-urile EMBED sau OBJECT. Compatibilitatea acestor tag-uri cu versiunile anterioare ale limbajului este limitată, putând apărea diferențe de afișare în diferitele navigatoare.

```
<embed src="cale/fisier.pdf" width="400" height="600">
```

```
<object data="cale/fisier.pdf" type="application/pdf" width="400" height="600">
      <p>Nu se poate afișa documentul pdf. <a href="cale/fisier.pdf"> Descarcă</a> fișierul.</p>
```

Formulare

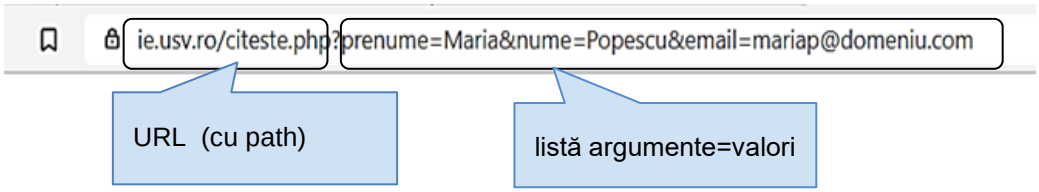
Formularele sunt mijlocul prin care utilizatorii pot trimite date la server prin intermediul paginilor Web.

Un formular HTML este încadrat între tag-urile `<form> </form>`.

Tag-ul de început conține două atribute obligatorii: *method* și *action*. „*method*” specifică metoda prin care se trimit datele la server și poate lua una din valorile POST și GET.

Metoda POST se folosește când se trimit date sensibile (precum parole, numere de card, adrese de email etc) sau volume mari de date (fișiere). Practic nu există limitări în volumul de date trimise prin această metodă, dacă serverul nu este configurat să aplice restricții.

Metoda GET trimite datele în formă de URL, vizibile în bara de adrese a navigatorului în forma:



Prima parte a adresei, până la simbolul ? este URL-ul la care se trimit datele din formular. El include numele complet al domeniului și calea (path) către fișierul care va prelua datele.

A doua parte, după simbolul ?, este o listă de perechi *nume=valoare*, separate de simbolul &. Datele trimise prin metoda GET sunt memorate de browser ca adrese web în istoricul navigării, putând fi devoalate cu ușurință.

Atributul *action* specifică URL-ul[4] la care se trimit datele.

```
<form action="http://domeniu.topdomeniu/cale" method="post">
```

[4] URL - Uniform Resource Locator. Un mecanism de identificare unică a unei resurse în spațiul Web. Are forma unei adrese Web

```
....
</form>
```

În funcție de modul concret de utilizare a formularului, tag-ul form poate conține și următoarele atribute opționale:

- *name,* este numele formularului, necesar pentru identificarea lui în cadrul documentului;
- *id*, un identificator unic pentru identificarea formularului
- *enctype*, specifică modalitatea de codare a datelor, când sunt trimise la server. Dacă nu se specifică se presupune a fi cea implicită, cu valoarea *application/x-www-form-urlencoded*. Pentru trimiterea fișierelor, valoarea este *multipart/form-data*.

Metoda GET are limitări în ceea ce privește lungimea datelor trimise și a URL-ului, limitări care depind de tipul de navigator dar și de configurația serverului. În general, lungimea adresei web, incluzând calea și datele, este aproape de 2 kB.

În interiorul formularului găsim următoarele elemente:

- Elemente de tip input:

sunt definite de eticheta nepereche <input> și includ: casete de text, casete de tip *password*, butoane radio, casete de selecție, câmpuri ascunse, câmpuri pentru fișiere, butoane.

  - *Casete de text*, afișează o casetă, pe un singur rând, în care utilizatorul poate introduce un text scurt. Admite câteva atribute printre care name, size și value care stabilesc numele casetei, lungimea ei – în caractere și valoarea inițială, pentru cazul în care caseta rămâne necompletată.
  - *Casete password*, sunt asemănătoare casetelor de tip text doar că la completare înlocuiește caracterele tastate de utilizator cu simbolul ●, pentru a nu putea fi citite de către alte persoane aflate în preajmă. Admite aceleași atribute ca și casetele de tip text.

    Sintaxa este, pentru ambele tipuri, similară:

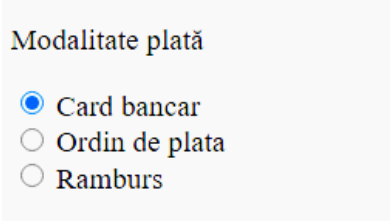
```
<input type="tip" name="nume_caseta" value="valoare">
```

  - *Butoane radio*, folosesc pentru selectarea unei singure opțiuni dintr-un grup de mai multe, incompatibile între ele. Atunci când o opțiune este selectată, cea selectată anterior se deselectează. Acceptă atributele *name*, *value* și *checked*. Toate butoanele din același grup

trebuie să aibă același nume. Valoarea atributului *value* a butonului selectat este trimisă la server la expedierea datelor. Atributul *checked* face ca butonul cu acest atribut să apară selectat. Un singur buton poate avea acest atribut. Exemplu:

```
<p>Modalitate plată</p>
<input type="radio" name="plata" value="card"
checked> Card bancar<br>
<input type="radio" name="plata" value="op"> Ordin
de plata<br>
<input type="radio" name="plată" value="ramburs">
Ramburs<br>
```

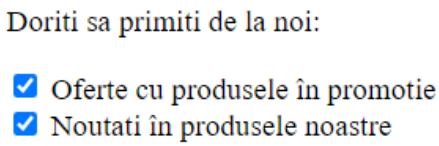
Rezultatul este cel din figura de mai jos:



- *Casete de selecție (checkbox)*, permit selectarea unei sau mai multor opțiuni dintr-o listă de opțiuni care nu sunt incompatibile între ele. Acceptă aceleași atribute ca și butoanele radio și respectă aceeași constrângere privind numele. Atributul *checked* poate apărea la mai multe casete din grup. Exemplu:

```
<p>Doriti sa primiti de la noi:</p>
<input type="checkbox" name="preferinta1"
value="oferte" checked> Oferte cu produsele în
promotie<br>
<input type="checkbox" name="preferinta2" value=
"noutati" checked> Noutati în produsele noastre<br>
```

Rezultatul poate fi văzut în figura următoare:



- *Câmpurile ascunse*, (hidden) nu sunt vizibile în document, dar sunt purtătoare de informații care sunt trimise la server. Sunt folosite de

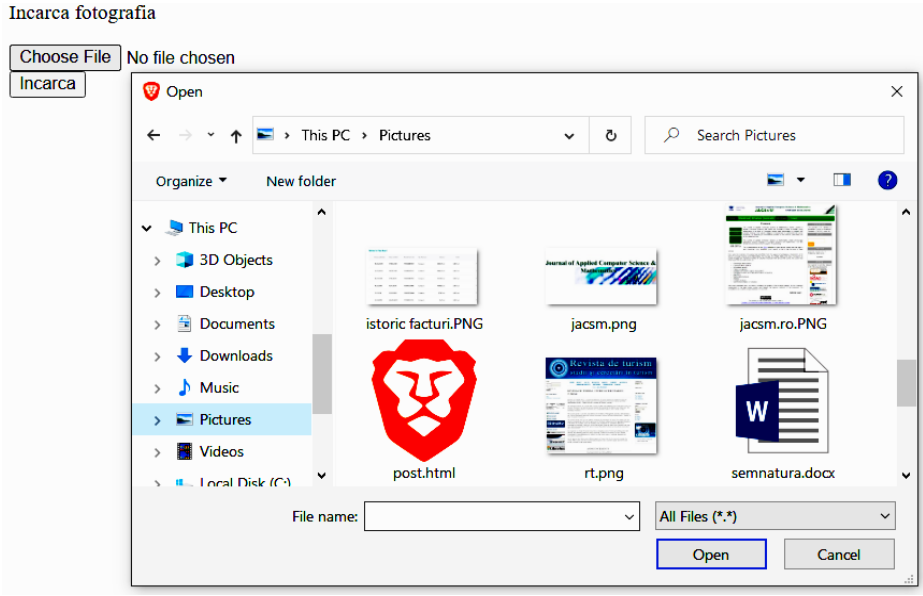
aplicațiile pentru Web pentru a stoca informații în pagini. Sintaxa este:

```
<input type="hidden" name="nume_camp" value="valoare">
```

- Câmpul *file*, este folosit pentru încărcarea unui fișier de la utilizator pe server. Metoda de trimitere trebuie sa fie *post* iar atributul *enctype* al etichetei *form* trebuie sa fie *multipart/form-data* (enctype="multipart/form-data"). Admite atributul *name* care atribuie un nume câmpului (valoarea lui este dată de numele fișierului selectat pentru încărcare) și lungimea câmpului, în caractere. Exemplu de utilizare:

```
<input type="file" name="nume_camp">
```

La apăsarea butonului *Choose File* se deschide exploratorul de fișiere pentru a da posibilitatea alegerii fișierului de transferat:



- Butonul *Submit*, este butonul care, la apăsare, declanșează trimiterea datelor din formular la server. Admite atributele name, value și size, care atribuie un nume butonului, determină textul scris pe buton și respectiv lungimea butonului. Exemplu:

```
<input type="submit" name="send" value="Incarca">
```

- Butonul *Reset*, readuce conținutul formularului la starea inițială. Admite aceleași atribute ca și butonul Submit. Sintaxa este similară

ca cea pentru butonul Submit, doar că atributul type are valoarea reset.

- Butonul simplu, *Button*, nu are o funcționalitate prestabilită, ea se atribuie prin program. Programele pentru Web, care se execută la client (în pagina Web) sunt scrise (JavaScript).

- **Liste de selecție**. Permit alegerea uneia sau mai multor opțiuni dintr-o listă. O listă de selecție este delimitată de etichetele `<select> </select>` și are un nume specificat de atributul *name*. Fiecare articol din listă este delimitat de eticheta `<option> </option>`. Fiecare opțiune are un atribut, *value*, care stabilește valoarea care se trimite la server când opțiunea este selectată. Textul dintre etichetele `<option> </option>` este cel pe care îl vede utilizatorul.

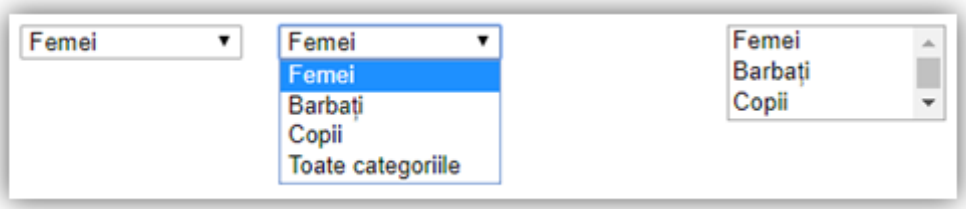
  În mod implicit lista afișează doar un articol, celelalte devenind vizibile la apăsarea butonului din extremitatea dreaptă a câmpului de selecție. Atributul *multiple* permite selectarea mai multor opțiuni, prin apăsarea butonului *Ctrl* în timpul selecției.

```
<select name="Categorii">
<option value="f">Femei</option>
<option value="b">Barbați</option>
<option value="c">Copii</option>
<option value="t">Toate categoriile</option>
</select>
```

  Dacă se dorește afișarea mai multor elemente ale listei se utilizează atributul *size* al etichetei *select*, ca în exemplul următor:

```
<select name="Categorii" size="3">
<option value="f">Femei</option>
<option value="b">Barbați</option>
<option value="c">Copii</option>
<option value="t">Toate categoriile</option>
</select>
```

De regulă, *size* se folosește împreună cu *multiple*. Rezultatul este cel din figura de mai jos:



Tag-ul `<optgroup>`, folosit în interiorul elementului `select` permite gruparea opțiunilor, ca în exemplul următor:

```
<p>Selecteaza un hobby</p>
  <select name="hobby" multiple size="6">
  <optgroup label="Sport">
<option value="Tenis">Tenis</option>
    <option
value="Fotbal">Fotbal</option>
  </optgroup>
  <optgroup label="Muzica">
    <option value="Rock">Rock</option>
    <option value="Jazz">Jazz</option>
  </optgroup>
</select>
```

Selecteaza un hobby

Sport
Tenis
Fotbal
Muzica
Rock
Jazz

Tag-ul `<optgroup>` are un singur atribut, *label*, iar valoarea acestuia stabilește numele grupului.

- **Arii de text (textarea)**. Sunt destinate introducerii de mesaje text, în volum mult mai mare decât permit casetele de text. Eticheta care definește aria de text este `<textarea>`. Este etichetă pereche și admite atributele *name*, *rows* și *cols*. Ultimele două stabilesc înălțimea – în rânduri și respectiv lățimea – în caractere, a suprafeței de text.

Elementele *input* de tip text și password, precum și *textarea* admit o serie de atribute opționale *disabled, readonly, minlength și maxlength*[5]. Atributul *disabled* dezactivează elementul iar conținutul lui nu poate fi modificat și nici nu este trimis la server. Spre deosebire de *disabled*, *readonly* deși nu permite editarea conținutului, trimite la server conținutul elementului. Atributele *minlength* și

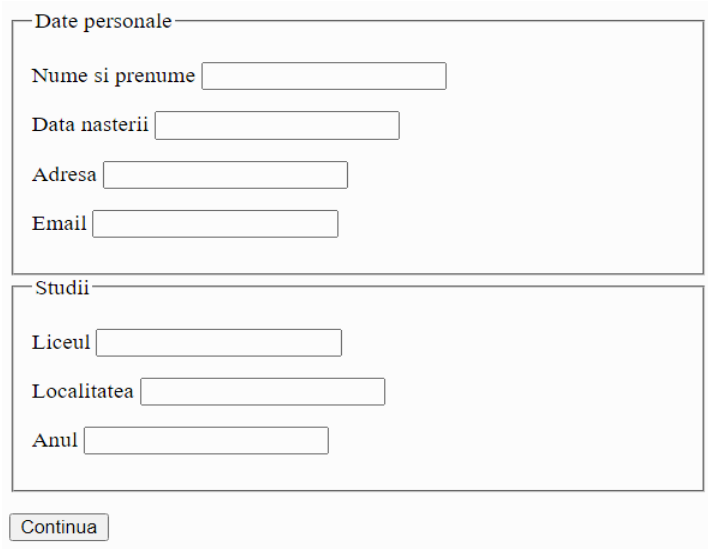
[5] Nu sunt singurele. Pentru detalii consultați https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes

maxlength stabilesc numărul minim și maxim al caracterelor admise, care vor fi trimise la server.

Elementele formularelor pot fi grupate pentru a oferi o experiență mai bună de utilizare. Pentru aceasta se folosesc tag-urile `<fieldset>` și `<legend>`. Ambele sunt tag-uri pereche: primul stabilește cadrul pentru elementele conținute iar al doilea adaugă o legendă grupului.

```
<fieldset>
  <legend>Nume grup</legend>
   <!-- elemente de formular -->
</fieldset>
```

Puteți consulta un exemplu de utilizare. Aspectul este cel din figura următoare:



În documentele complexe se introduc comentarii, care ajută la citirea codului de către proiectantul paginii dar nu trebuie afișate de navigator. De asemenea, e posibil să dorim ca anumite fragmente de cod HTML din sursă să nu fie reproduse de navigator. Comentariile sau liniile de cod care nu trebuie reproduse, se marchează astfel:

<!- - comentariu sau cod care trebuie ascuns -->

Tag-uri noi HTML5

Tag-uri pentru formulare

HTML5 vine cu câteva tag-uri și atribute care îmbogățesc galeria de tag-uri HTML 4 și aduc un plus de funcționalitate.

Un prim atribut, care se aplică elementelor `<input>`, `<select>` și `<textarea>` și care împiedică trimiterea datelor până când câmpurile respective nu sunt completate în formatul corespunzător tipului de câmp, este *required*. Exemplu: `<input type="text" required>`

Un alt atribut util este *placeholder*, care afișează un mesaj în câmpurile necompletate. Mesajul dispare când se completează câmpul respectiv:

```
<input type="text" name="numele" placeholder="Numele tău">
```

Se aplică elementelor *input* de tip text, password și elementului *textarea*. Conținutul *placeholder* nu este trimis la server ca valoare a câmpului.

Atributul *autofocus* determină câștigarea focusului (selectarea) pe elementul care deține atributul. Acest atribut nu este foarte „de încredere”, deoarece nu funcționează întotdeauna conform așteptărilor[6] și nici nu este suportat de toate navigatoarele[7]. Atributul este, adesea, înlocuit de un mic fragment de cod JavaScript pentru a realiza același lucru.

Atributul *autocomplete* servește la activarea/dezactivarea autocompletării elementelor de formular pe baza istoricului. Implicit, *autocomplete* este *on* (dacă nu este specificat). Pentru dezactivare, valoarea este *off*:

```
<input type="text" name="numele" autocomplete="off">
```

Atributul *list* funcționează împreună cu tag-ul `<datalist>` pentru a furniza o listă de valori predefinite pentru un element *input* de tip text:

```
<p>Alege navigatorul preferat:</p>
<input type="text" list="browsers">
<datalist id="browsers">
    <option value="Firefox">
    <option value="Chrome">
    <option value="Internet Explorer">
    <option value="Opera">
    <option value="Safari">
```

[6] https://bugs.chromium.org/p/chromium/issues/detail?id=121006

[7] https://www.lambdatest.com/autofocus-attribute

```
</datalist>
```

El indică lista *datalist* care se asociază cu elementul *input*.

Între elementele nou introduse sunt câteva subtipuri ale elementului *input* de tip text:

number: admite numai valori numerice. Admite atributele *min* (stabilește valoarea minimă acceptată), *step* (stabilește pasul cu care se poate modifica valoarea elementului) și *value* (stabilește valoarea inițială).

```
 <input type="number" min="5" step="0.1" value="8">
```

email: admite adrese de email. Verifică prezența simbolului @ astfel că o adresa de tipul *user@somwhere* este considerată validă. Din acest motiv, tipul email se folosește împreună cu atributul *pattern*. Acesta stabilește, cu ajutorul expresiilor regulate, un șablon pe baza căruia este validată valoarea introdusă de utilizator:

```
Email:    <input    type="email"    id="email"    name="email"
pattern="[a-z0-9.]+@[a-z0-9.-]+\.[a-z]{2,}$">
```

În exemplul de mai sus, adresa de email trebuie să fie în forma *user@domeniu.top*, unde:

- *user* poate fi orice șir de caractere care conține literele a-z, cifrele 0-9 și simbolul punct;
- *domeniu* este orice șir compus din aceleași simboluri ca și *user* iar, în plus, este acceptat simbolul -
- *top* este un șir literal de minim 2 caractere.

Pentru documentare asupra expresiilor regulare, puteți consulta un material scris într-un limbaj accesibil:

https://marplo.net/php-mysql/expresii_regulate_ereg.html.

tel: admite un șir care are forma unui număr telefonic. Pentru că convențiile de scriere a numerelor de telefon diferă de la țară la țară, utilizarea lui este limitată:

```
<input  type="tel"  id="myphone"  placeholder="xxxx-xxx-xxx"
required>
```

Și în acest caz utilizarea expresiilor regulate poate duce la o îmbunătățire a validării conținutului.

date: așteaptă o dată calendaristică în format aaaa-ll-zz (yyyy-mm-dd). Afișează un calendar din care se poate selecta interactiv o anumită dată:

```
Selectează data: <input type="date" value="2020-12-09" id="mydate">
```

range: afișează un cursor cu ajutorul căruia utilizatorul poate regla o valoare. Se utilizează împreună cu atributele *min*, *max* și *step*, care stabilesc limitele domeniului și pasul cu care se modifică valoarea când se deplasează cursorul. Atributul value stabilește valoarea inițială.

color: permite selecția unei culori:

```
Selectează culoarea: <input type="color" value="#00ff6e">
```

Puteți examina un exemplu cu *date range* și *color* aici https://playcode.io/713334/.

Alte tipuri de elemente *input*: **time**, **week**, **month**, **datetime-local**, **search**, **URL**[8].

Alte tag-uri introduse de HTML5

Tag-urile introduse de HTML5 sunt tag-uri semantice, deoarece numele lor desemnează rolul pe care îl au în document.

Tag-uri semantice. Prin denumirea lor desemnează rolul pe care îl au în cuprinsul documentului:

```
<article>
<aside>
<details>
<figcaption>
<figure>
<footer>
<header>
<main>
<mark>
<nav>
<progress>
<section>
```

[8] https://developer.mozilla.org/en-US/docs/Learn/Forms/HTML5_input_types

<summary>

<time>

Elemente precum <header>, <nav>, <section>, <article>, <aside> și <footer> acționează similar cu elemente . Ele grupează alte elemente împreună în secțiuni de pagină. Dacă o etichetă ar putea conține orice tip de informație, ne putem imagina ce fel de informații poate conține eticheta semantică <header>.

În lipsa acestora, elementele erau (și încă se mai folosește) înlocuite cu elemente care aveau un identificator, cu rol multiplu: indică rolul secțiunii în cadrul documentului, permite aplicarea de stiluri diferențiate și asigură o metodă de acces la element prin intermediul limbajelor de programare pentru Web. Exemple:

```
<div id="article"> </div>
<div id="aside"> </div>
<div id="footer"> </div>
```

Tag-urile <section> și <article> nu au vreo redare specială și, în plus, sunt interschimbabile: o secțiune poate conține mai multe articole și invers, un articol poate conține mai multe secțiuni. Exemplul de la adresa https://playcode.io/711312/ arată cum, pe cele două coloane rolurile elementelor se inversează, dar aspectul vizual este identic.

Un exemplu de utilizare a altor tag-uri semantice, fără nicio stilizare poate fi văzut la adresa: https://playcode.io/711285/

Un exemplu de utilizare a acelorași tag-uri, cu stilizarea elementelor, este disponibil la adresa: https://playcode.io/711297/

Tag-ul <figure> permite încadrarea imaginilor într-un element de tip bloc, cu un format prestabilit (marginile de jur împrejur). Un exemplu de utilizare este vizibil la adresa https://playcode.io/711330/. Pentru o mai bună încadrare în pagină, cele două tag-uri au primit stiluri diferite, prima aliniată la stânga și a doua la dreapta.

<figcaption> adaugă o legendă figurii.

Tag-ul <mark> marchează porțiunea de text încadrată, așa cum o face markerul.

Tag-ul <progress> este folosit pentru sugera progresul unei activități online (de exemplu timpul scurs sau numărul de întrebări rezolvate dintr-un chestionar. Cel mai adesea este utilizat cu scripturi JavaScript. Puteți un exemplu de utilizare accesând acest link.

Tag-urile <details> și <summary> se folosesc împreună, în forma:

```
<details>
  <summary>HTML Semantic</summary>
  <p>Many HTML tags have semantic meaning. That is, the
element itself conveys some information about the type of
content contained between the opening and closing tags.
  </p>
</details>
```

Exemplul următor ilustrează un meniu armonică folosind tag-urile `<details>` și `<summary>`.

Limbajul CSS

Introducere

Lipsindu-i foarte multe din atributele versiunilor anterioare, care ajutau la o formatare acceptabilă a documentelor, versiunea 5 a limbajului HTML5 are posibilități foarte reduse de formatare a unui document. Aspectul documentelor existente astăzi pe Web este foarte atractiv și se bazează pe un limbaj complementar limbajului HTML, care permite stabilirea cu precizie a caracteristicilor elementelor, de la culoare, comportament și până la poziționare.

Limbajul, numit CSS (Cascade Style Sheets – foi de stiluri în cascadă), se află la versiunea a treia, de unde și numele CSS3. Limbajul constă într-un set de directive *proprietate:valoare*, asociat elementelor din document. Prin felul în care a fost gândit, limbajul asigură o separare a conținutului paginii față de definițiile de stil, fapt care conferă o bogăție semantică sporită documentului, făcându-l mai ușor de citit și indexat de către motoarele de căutare. Totodată se realizează o modalitate flexibilă de stabilire a aspectului documentului, fiind foarte simplu de făcut ca același conținut să fie reprodus în moduri diferite.

Există trei modalități de integrare a directivelor de formatare în cadrul documentului:

A. În fișier extern. Un fișier distinct, cu extensia css, conține toate definițiile de stil. În documentul HTML, o etichetă <link> face legătura cu fișierul css. Modificarea definițiilor din fișierul extern afectează toate documentele care folosesc respectivele definiții. Un document HTML poate include mai mult de un fișier de definiții css.
B. În antetul documentului. În secțiunea <style> </style> se includ directivele de formatare pentru elementele din document. Aceste directive afectează doar pagina curentă și prevalează asupra definițiilor din fișierul extern. Altfel spus, dacă un element este definit printr-un fișier css extern, iar în antetul documentului se modifică unele dintre aceste definiții, elementul va fi reprodus conform definițiilor modificate. Definițiile care nu au fost modificate sunt aplicate ca atare.
C. În linie cu eticheta (inline). Directivele se aplică în interiorul etichetei elementului, cu ajutorul atributului style, după sintaxa:

```
<element style="proprietate1:valoare1; proprietate2:
valoare2;..">.
```

Definițiile *inline* prevalează asupra definițiilor din antet. Acest ultim mod trebuie evitat deoarece nu realizează separarea dorită dintre conținut și definiții.

Selectori

Pentru definirea stilurilor în fișier extern și în antet se folosesc selectorii, care indică elementele care sunt afectate de definiții, după modelul:

```
selector {
  proprietate1:valoare1;
  proprietate2: valoare2,
.
.
}
```

Selector este orice etichetă validă pentru tipul de document declarat (p, h, div, li, td, a etc) dar și *identificatori* ai elementelor sau *clase*.

Perechile *proprietate:valoare* se separă prin punct și virgulă ;. Proprietățile diferă în funcție de selector, nu toți selectorii acceptă toate proprietățile.

Identificatori și clase

Dacă stabilim o serie de proprietăți pentru selectorul p (paragraf), atunci toate paragrafele dintr-un document vor fi afectate. Rareori acest lucru este de dorit, cel mai adesea dorim ca doar unul sau doar câteva paragrafe să fie afectate, restul păstrându-și aspectul implicit sau altul pe care să-l putem stabili independent de al celorlalte. Aici intervin identificatorii sau clasele.

Unele elemente dintr-un document pot avea identificatori, care se atribuie de forma:
`<p id="nume">`

Numele identificatorului trebuie să fie unic în document. Rolul lui este, pe de o parte, să permită atribuirea unor definiții de stil dedicate doar elementului respectiv și, pe de altă parte, pentru a facilita accesarea elementului în programele JavaScript.

Unui element care are un identificator i se pot stabili proprietățile pe baza identificatorului, după aceeași regulă generală, doar că selectorul este format din numele identificatorului precedat de simbolul #. Spre exemplu, în documentul ce mai jos, care conține două paragrafe, doar paragraful „p1” este afectat de proprietățile de stil:

```
<style>
#p1 {
      proprietate1: valoare;
      proprietate2: valoare;
}
</style>
<body>
```

```
<p id="p1">Paragraf care este afectat de proprietatile
proprietate1 si proprietate2</p>
<p>Paragraf cu comportament implicit</p>
</body>
```

Dacă dorim ca mai multe elemente să fie afectate de același set de proprietăți, grupăm elementele în clase, cu ajutorul atributului *class*, adăugat etichetelor alimentelor ce urmează a fi afectate:

```
<p class="titlu">Paragraf cu proprietatile clasei titlu</p>
<p class="rezumat">Paragraf cu proprietatile clasei
rezumat</p>
<p class="titlu">Paragraf cu proprietatile clasei titlu</p>
```

Pentru definirea stilului claselor se folosește aceeași sintaxă ca pentru selectori, cu deosebirea că în locul numelui selectorului se scrie numele clasei precedat de semnul punct (.):

```
.titlu {
     proprietate1: valoare;
     proprieatate2: valoare;
}
.rezumat {
     proprietate3: valoare;
     proprietate4: valoare;
}
```

Exemplu de utilizare identificatori și clase

Dacă o serie de proprietăți sunt comune mai multor clase sau identificatori, ele se pot atribui prin enumerarea claselor sau identificatorilor în definirea proprietăților, urmând ca pentru fiecare clasă sau identificator să se scrie doar proprietățile distincte. În loc de:

```
.titlu {
     proprietate1: valoare_comuna;
     proprieatate2: valoare;
}
.rezumat {
     proprietate1: valoare_comuna;
     proprieatate4: valoare;
}
```

unde „proprietate1” are aceeași valoare pentru ambele clase, putem scrie

```
.titlu, .rezumat {
      proprietate1: valoare_comuna;
}
.titlu {
      proprieatate2: valoare;
}
.rezumat {
      proprieatate4: valoare;
}
```

Un element poate avea proprietăți comune mai multor clase și atunci numele claselor se trec ca valori ale atributului class, fără virgulă între numele claselor:

```
<p id="prim-paragraf" class="titlu text-centrat">Paragraf cu proprietatile clasei titlu</p>
```

unde titlu și text-centrat sunt două clase diferite, iar paragraful va avea caracteristicile definite pentru identificatorul prim-paragraf și cele două clase.

Comentariile, ca și directivele care trebuie ignorate de navigator, se marchează astfel:

```
/* comentariu: proprietate 2 va fi ignorata*/
h3 {
  proprieatate1: valoare;
/* proprieatate2: valoare; */
}
```

Exemplu de utilizare proprietăți comune si comentarii

Selectori combinați

Selectarea descendenților se face folosind sintaxa

```
selector selector-descendent {proprietate:valoare;}
```

Exemple:

```
div p {background-color:yellow} selectează toate paragrafele din orice div.
#d1 p { background-color:yellow} selectează toate paragrafele din elementul cu id ‘d1’
.foo > p selectează toate paragrafele care sunt copii ai elementului din clasa “foo”.
```

Spre exemplificare, în codul următor doar primul și al treilea paragraf sunt copii ai elementului div din clasa “foo”. Al doilea paragraf fiind inclus în alt tag nu este copil al div-ului din clasa amintită. Exemplu online.

```
<div class="foo">
      <p>Primul paragraf din div clasa "foo"</p>
      <section>
        <p>Al doilea paragraf nu este copil al div-ului din
clasa "foo"</p>
      </section>
     <p>al treilea paragraf din div clasa "foo"</p>
 </div>
```

Proprietăți pentru text

Fontul este designul caracterelor alfanumerice. Fonturile se caracterizează prin nume – care identifică un anumit design, dimensiune și stil (obișnuit, oblic, îngroșat, evidențiere - cu tag-ul *mark*).

Paragraful este o parte de text din document, separată de alte părți de text, printr-un spațiu. Paragrafele se caracterizează prin aliniere, spațierea liniilor de text, indentări, efecte speciale (sublinieri, tăieri cu linie simplă sau dublă).

Stiluri pentru font și paragrafe

Toate elementele care conțin text, cum sunt body, p, div, h, a etc. admit următoarele proprietăți pentru text:

proprietate	valoare	descriere
font-family	arial, tahoma, helvetica, "times new roman", serif	familia de fonturi folosită pentru text
font-weight	normal, bold, bolder, lighter	grosimea caracterelor
font-size	n (valoare întreagă, în px, pt sau em)	înălțimea font-ului
font-style	normal, italic	înclinarea textului
text-decoration	none, underline, overline, line-through	text subliniat, linie deasupra sau tăiat
text-transform	none, lowercase, uppercase, capitalize	transformă în minuscule, majuscule, prima literă din cuvânt în majusculă

text-align	left, right, center, justify	alinierea textului
text-indent	n (valoare întreagă, în px, pt sau em)	aliniază prima linie a unui paragraf
line-height	normal, number (1, 1.5), length (12pt, 1.5em)	Stabilește distanța dintre rândurile de text

Un exemplu de aplicare a acestor proprietăți poate fi văzut la https://playcode.io/717024/. Modificați valoarea proprietății *text-decoration* în *line-through* și a *line-height* în *1.5*. Notați rezultatul.

Pe lângă setul uzual de fonturi, se pot importa fonturi de pe site-uri care oferă această funcție în mod gratuit. Cel mai popular site este fonts.google.com. Procedura este foarte simplă și oferă un set bogat de fonturi care pot înfrumusețea aspectul paginilor. Un ghid de utilizare este oferit chiar de Google.

Culori

Culorile se aplică atât textului cât și fundalului acestuia. Se exprimă fie în format numeric (zecimal sau hexazecimal) sau literal. Pentru exprimarea numerică se specifică ponderea fiecărei culori în formatul RGB (Red, Green, Blue), ex: #ffccaa - hexazecimal, rgb(255,204,170) - zecimal. Pentru exprimarea literală se folosesc numele culorilor, în engleza americană: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white, yellow etc. De câte ori se modifică, prin stil, culoarea implicită a fontului, se recomandă să se schimbe și culoarea fundalului pentru a asigura contrastul necesar unei bune lizibilități. Proprietatea pentru culoarea textului este `color`, iar pentru fundal: `background-color`.

Exemple: https://playcode.io/717047/

Modelul box

Există două tipuri de comportamente pentru elementele HTML: *inline* și *block* (bloc). Exceptând elementul *img*, toate celelalte elemente de tip inline nu posedă proprietăți prin care să li se stabilească lățimea, înălțimea și margini superioare și inferioare precum și distanțele marginilor superioare și inferioare față de conținut.

Aceste proprietăți se regăsesc la elementele cu comportament de tip block, ca în figura alăturată. Elementele de tip block se comportă ca și când ar fi precedate și urmate de un tag BR, plasându-se întotdeauna pe rânduri noi.

Comportamentul poate fi modificat cu ajutorul proprietății *display*, care poate avea valorile: *block*, *inline, inline-block, none* și *flex*[9]. Elementele cu proprietatea *inline-block* se comportă ca elementele de tip *inline* doar că permit stabilirea proprietăților care lipsesc în mod natural acestui tip de elemente.

Valorile acestor proprietăți se exprimă în diverse unități de măsură care pot fi grupate în două mari categorii: absolute și relative.

Unități de măsură absolute (selecție dintre cele mai utilizate)

Unitatea de măsură	Descriere
mm	milimetri
pt	puncte (1 pct=1 inch/72 ≅2,54 cm/72)
px	pixel
pc	picas (1 pc=12pt)

Unități de măsură relative (selecție dintre cele mai utilizate)

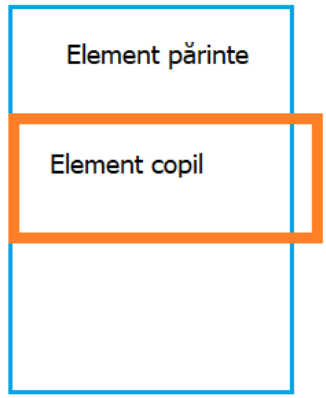
Unitatea de măsură	Descriere
em	Relativ la mărimea fontului sau elementului părinte (1.5em înseamnă de 1,5 ori mărimea fontului curent)
rem	(root em) - relativ la mărimea elementului rădăcină
%	procent din elementul părinte
vw	1% din lățimea viewport-ului[10]
vh	1% din înălțimea viewport-ului.

[9] Se va discuta la Sistemul flexbox.

[10] viewport: dimensiunea ferestrei navigatorului

Exemple: https://playcode.io/717067/

Mai multe informații, aici.



În mod implicit, lățimea și înălțimea atribuite unui element se aplică numai casetei de conținut a elementului. Marginile, grosimea chenarului sau căptușeală, dacă există, ele se adăugă la lățime și înălțime pentru a ajunge la dimensiunea casetei redate pe ecran. Aceasta înseamnă că, atunci când stabilim lățimea și înălțimea, trebuie să ținem cont valorile adăugate astfel încât designul paginii să nu se modifice.

Imaginea alăturată prezintă două elemente de tip bloc cu aceeași lățime, unde elementul copil este inclus în elementul părinte. Elementul copil are un chenar mai gros și *padding* adăugate, ceea ce face ca elementul copil să nu încapă, pe orizontală, în elementul părinte. Exemplul poate fi vizionat online.

Umplutura (padding)

este spațiul cu care se înconjoară conținutul (content).

Poate fi definită individual, pe fiecare parte, sau global, identic pentru toate părțile:

- padding: 4px 5px 2px 10px (4 px sus, 5 px dreapta, 2 px jos și 10 px stânga.
- padding: 4px (4 px de jur împrejur)
- padding-top: 0.5em; padding-bottom: 0.2em (modifică doar umplutura sus și jos)

Chenarul (border)

este definit prin grosime, culoare și aspect. Este exterior umpluturii și poate fi definit global sau individual, pentru fiecare caracteristică în parte:

`border: 2px solid silver` (chenar de grosime 2px, aspect plin, culoare argintie, de jur împrejur)

Valorile pentru proprietățile aspectului sunt; *solid*, *dotted*, *dashed*, *double*, *groove*, *ridge*, *inset* și *outse*t. Proprietatea pentru aspect este *border-style*: border-style: dashed (chenar linie întreruptă).

Proprietatea pentru grosime este *border-width*, iar pentru culoare *border-color*. Pentru stabilirea individuală a grosimii și culorii chenarului, pentru fiecare latură, se folosesc proprietățile: *border-top-width*, *border-right-width*, *border-bottom-width* și *border-left-width*, respectiv *border-top-color*, *border-right-color*, *border-bottom-color* și *border-left-color.*

Chenarul poate avea marginile rotunjite. Proprietatea este border-radius iar valoarea exprimată în oricare dintre unitățile de măsură enumerate, reprezintă raza. Se poate aplica global: *border-radius: 5px*, global cu valori diferite *border-radius: 5px 8px 3px 4px,* unde prima valoare corespunde colțului stânga-sus, respectiv se pot aplica individual, pe fiecare colț: *border-top-right-radius*, *border-top-left-radius*, *border-bottom-right-radius*, *border-bottom-left-radius*.

Marginile (margin)

reprezintă spațiul dintre exteriorul chenarului și elementele vecine. Se pot defini pentru toate laturile într-o singură directivă sau separat, pentru fiecare latură:

- margin: 10px - margine egală, 10 px, de jur împrejur;
- margin: 10px 5px 10px 0px - margine 10 px sus, 5 px dreapta, 10 px jos și 0 px stânga;
- margin-left: 3px - margine stânga: 3 px;

Este mai comod să se stabilească dimensiunile finale ale elementelor iar tot ce se adaugă, să se adauge spre interior.

Navigatoarele interpretează diferit proprietățile pentru box-uri, în special dacă lipsește declarația de tip de document sau aceasta este incorectă. De aceea este recomandat să se precizeze în mod explicit modul de interpretare, cu atributul *box-sizing*. Valoarea *border-box*, determină navigatorul să includă umpluturile și grosimea chenarului în interiorul dimensiunilor stabilite pentru elementul de tip box. Valoarea *content-box* adaugă aceste valori la valoarea stabilită a dimensiunilor, Figura 24.

O directivă CSS de forma:

```
*  {box-sizing: border-box}
```

va face ca toate elementele de tip box să comporte ca în Figura 24a.

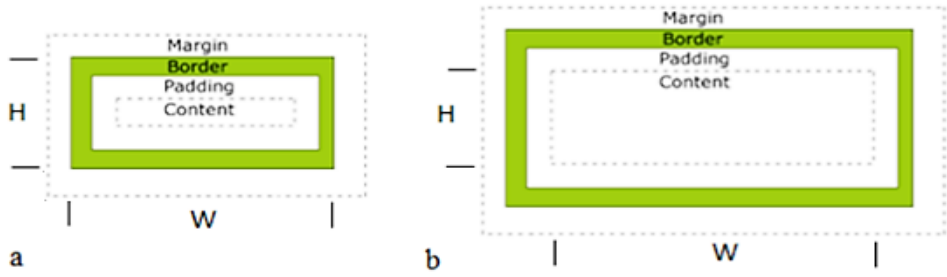


Figura 24. Comportamentul navigatoarelor pentru valorile border-box și box-content

Adăugați în codul CSS de la exemplul https://playcode.io/717100/ declarația

```
*  {box-sizing: border-box}
```

și notați rezultatul. [11]

Fundalul

(background) poate fi aplicat interiorului unui element de tip box sub formă de culoare sau imagine. Proprietățile CSS pentru background sunt:

- *background-color*; stabilește culoarea de fundal, ex: `background-color: #c0c0c0`
- *background-image*; stabilește imaginea folosită pentru fundal, ex: `background-image: url("blue_velvet.gif")`, unde *blue_velvet.gif* este numele fișierului imagine
- *background-repeat*; stabilește dacă imaginea de fundal se repetă și pe ce direcție. Are valorile: *no-repeat* (nu se repetă), *repeat-x* (repetare pe orizontală), *repeat-y* (repetare pe verticală). Valoarea implicită este repetare pe orizontală și verticală. Ex: `background-repeat: repeat-x repeat-y`;
- *background-attachment*; specifică dacă imaginea de fundal este fixă în raport cu conținutul sau se deplasează odată cu acesta. Valorile sunt *fixed* și *scroll*, valoarea implicită este *scroll*. Ex: `background-attachment: fixed`;

[11] Nu se recomandă această sintaxă deoarece solicită browser-ul în identificarea tuturor elementelor paginii. Se recomandă să se enumere toate elementele cu acest comportament: h1, h2, h3, p, div, main, #container, section, article, header {box-sizing: border-box}

- *background-position*; specifică poziția unde apare imaginea de fundal. Are valorile: *top*, *bottom*, *left*, *right*. Exemplu: `background-position: top right`.

Exemple de utilizare a proprietății *background-repeat* se pot vedea aici. Rezultatul este ca în Figura 25.

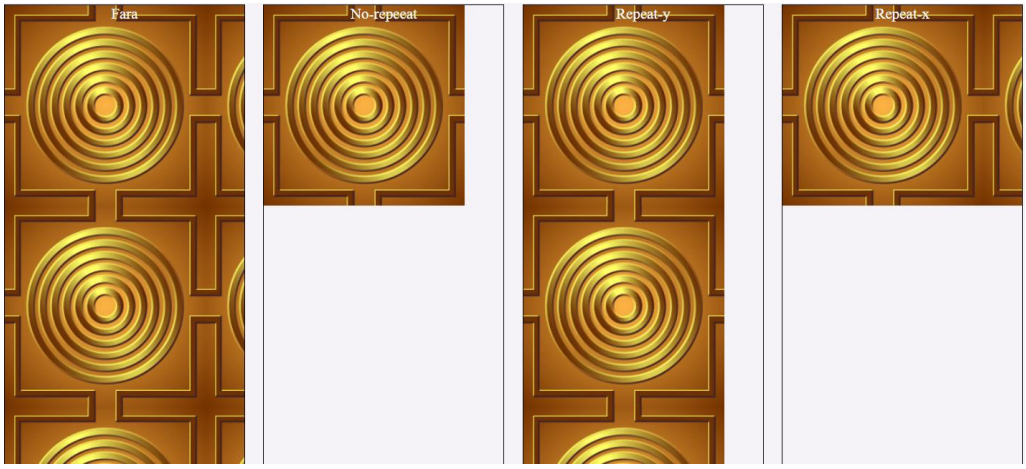


Figura 25. Utilizarea proprietății background-repeat

Dimensiunea fundalului

Dimensiunea fundalului se poate modifica cu proprietatea *backround-size*. Valorile pe care le poate lua sunt: auto (valoare implicită), *length*, *percentage*, *cover* și *content*.

Length și *percentage* stabilesc lățime și înălțimea în valori absolute (px, em, rem) sau procentuale relativ la dimensiunea elementului părinte. De obicei se dau două valori, prima reprezentând lățimea iar a doua înălțimea. Dacă se specifică doar o valoare, aceea este lățimea iar înălțimea se potrivește automat pentru păstrarea raportului de formă. Ex:

```
background-size: 300px 150px;
background-size: 300px;
background-size: 100% 50%;
background-size: 100%;
```

Valoarea *cover* întinde imaginea astfel încât toată suprafața box-ului să fie acoperită. Valoarea *content* redimensionează imaginea astfel încât aceasta să fie complet vizibilă în interiorul elementului. Figura 26 ilustrează modul de afișare a

imaginii de fundal care are dimensiunea originală de 250px X 250px într-un box cu dimensiunile de 400px X 300px.

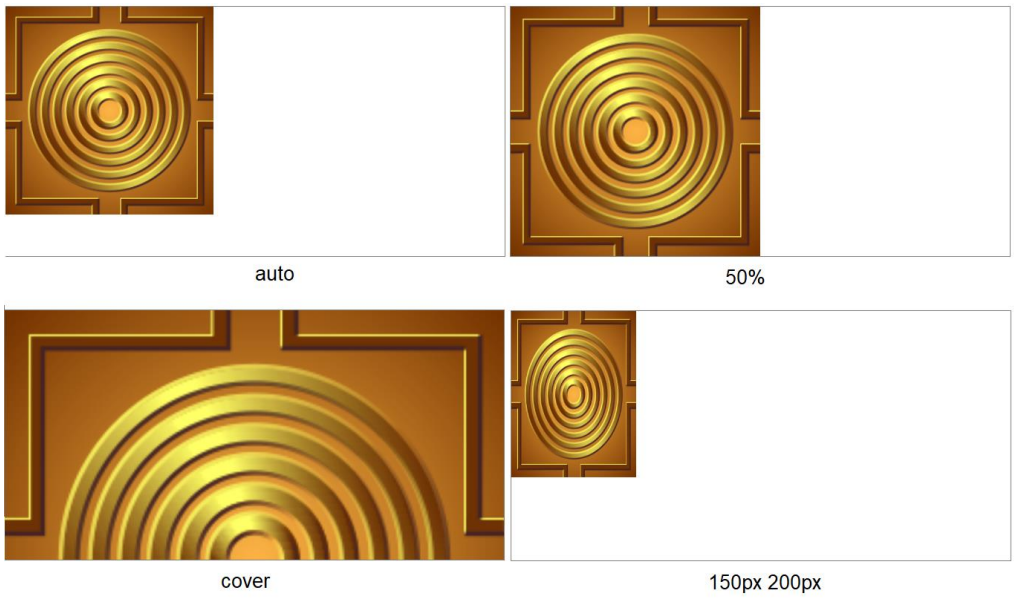


Figura 26. Diverse aspecte ale fundalului pentru valori diferite ale proprietății background-size

Aici aveți un exemplu în care puteți vedea ușor rezultatul schimbând valoarea proprietății din *cover* în *auto*, apoi *content* și respectiv 100%.

Stiluri pentru liste

Listele pot fi personalizate cu ajutorul atributelor:

- *list-style-type*; stabilește tipul de marcator sau enumerator al listei: *disc*, *circle*, *square*, *upper-roman*, lower-alpha, *none* etc. Vezi și https://developer.mozilla.org/en-US/docs/Web/CSS/list-style-type
- *list-style-position*; specifică dacă marcatorul/enumeratorul listei apare în interiorul sau în exteriorul elementului listei, Figura 27. Are două valori, *inside* și *outside*
- *list-style-image*; înlocuiește marcatorul cu o imagine specificată ca argument, ex: `list-style-image: url("o_bila.gif");`

De asemenea, sunt utile proprietățile *margin* și *padding* pentru stabilirea alinierii listei.

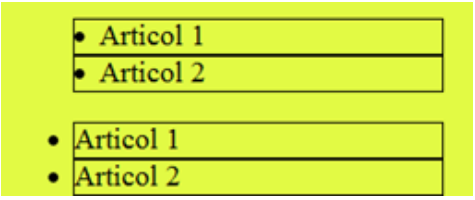
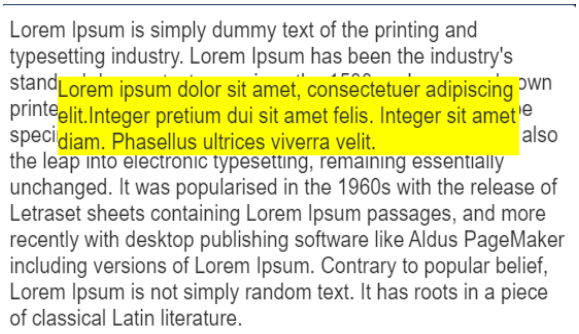


Figura 27. Listă cu marcatorii inside și outside

Poziționarea elementelor de tip box



În mod normal elementele sunt poziționate în ordinea normală a derulării paginii, unul după altul. Este așa numita poziționare statică. Cu ajutorul CSS poziționarea elementelor de tip box se poate face în alte câteva moduri: fix, absolut și relativ.

Atributul pentru stabilirea tipului de poziționare este *position*, care poate avea următoarele valori: *static*, *fixed*, *absolute* și *relative*. În modul fix (fixed) elementul are o poziție fixă față de fereastra navigatorului și nu se mișcă nici la derularea ferestrei. Poziția efectivă se stabilește cu atributele *top*, *left* și *right*, având valori numerice exprimate în una din unitățile de măsură acceptate. Ex:

```
<style>
div {
position: fixed;
top:50px;
left: 100px;
width: 250px;
background-color:yellow;
}
</style>
<body>
<div> Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Integer pretium dui sit amet felis. Integer sit amet diam. <p>Phasellus ultrices viverra velit.</div>
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but
```

```
also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.
Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature.</p>
</body>
```

Vizualizati exemplul, redimensionând fereastra navigatorului până apare bara scroll verticală. Derulați sus-jos și observați poziția div-ului galben.

În modul absolut (absolute), elementul este șters din poziția în care ar apărea în mod normal și dus în poziția indicată de atributele *top*, *left* sau *right*. Poziția absolută se raportează la elementul părinte; dacă elementul părinte este o secțiune (strat, div), poziția va fi față de marginile de sus, stânga sau dreapta ale părintelui.

În modul relativ (relative), elementul este deplasat față de poziția în care s-ar afla în mod normal, cu distanțele precizate de atributele top, left sau right. Exemplul următor produce rezultatul din Figura 28a, în care stratul „text” curge normal, după paragraf. Dacă se decomentează atributele *top* și *left*, rezultatul este cel din Figura 28b, în care startul este deplasat la stânga cu 20 px și în sus cu 10 px.

```
<style>
#text {
      position: relative;
      /*top:-10px;
      left: -20px;*/
      width: 250px;
      background-color:yellow;
}
p {
      background-color:black;
      color:white;
}
</style>
<body>
<p>un paragraf</p>
<div id="text"> Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Integer pretium dui sit amet felis. Integer sit amet diam. Phasellus ultrices viverra velit.</div>
</body>
```

Tot pentru poziționare este util atributul *float.* Are valorile *left*, *right* și *none*. Specifică poziția în care se amplasează elementul, față de elementul părinte, permițând înconjurarea lui cu text pe partea opusă. Valoarea *none* stabilește comportamentul implicit, cel în care elementul nu este înconjurat de text.

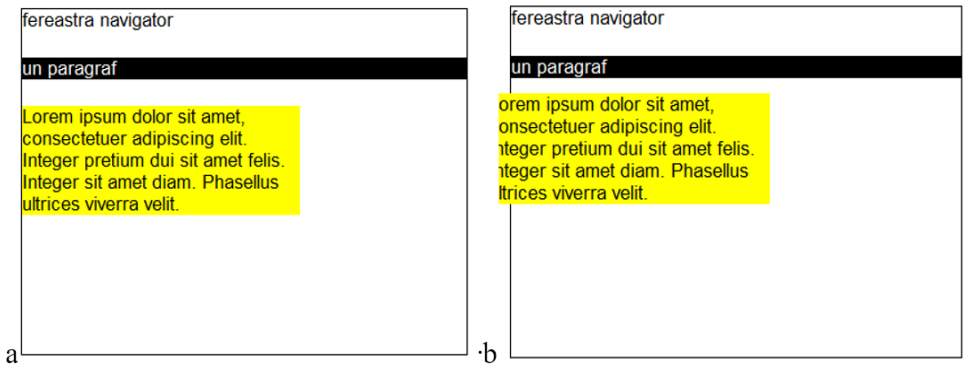


Figura 28. Poziționarea relativă, a) înainte b) după

Următoarea secvență de cod produce rezultatul din Figura 29, unde cele două straturi au setat atributul *float* cu valoarea *right*.

```
<style>
.text {
      float: right;
      margin:  5px;
      width: 200px;
      background-color:yellow;
}
</style>
<body>
<div id="container">
<p>fereastra navigator</p>
<div class="text"> Si acest element are atributul float cu valoarea right.</div>
<div class="text"> Acest element are atributul float cu valoarea right.</div>
<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. </p>
</div>
</body>
```

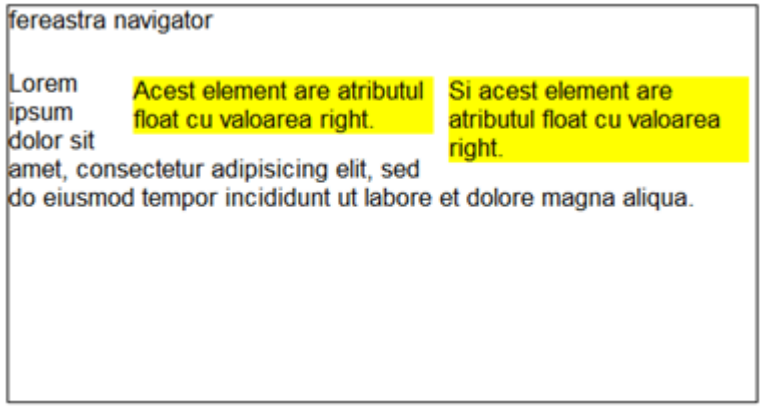


Figura 29 Două straturi cu atributul float:right

Pentru a anula plutirea elementelor și a putea continua poziționarea altor elemente care urmează, se utilizează proprietatea *clear*, care specifică pe ce părți ale unui element nu se permite elementelor plutitoare să plutească. Proprietatea *clear* poate lua una dintre valorile *left*, *right* sau *both*. Un exemplu poate fi văzut la adresa https://playcode.io/717886/ și are aspectul din Figura 30.

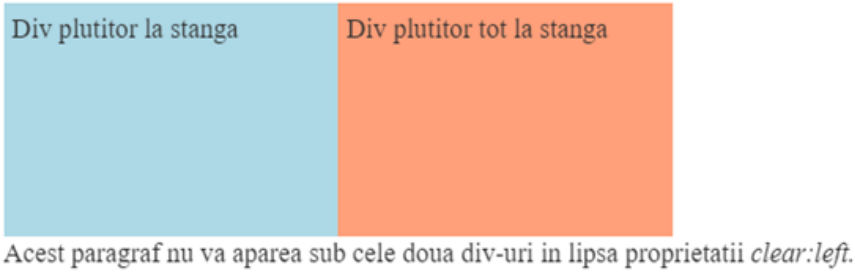


Figura 30. Utilizarea proprietății clear *pentru anularea proprietății* float

1. Eliminați atributul *clear* din paragraful care urmează după cele două div-uri și observați rezultatul.
2. Adăugați înapoi atributul. Modificați lățimea ferestrei și observați ce se întâmplă cu cele două div-uri.
3. Readuceți fereastra la dimensiunea inițială. Modificați valoarea atributului *float* pentru div-ul *d2* din *left* în *right* și observați ce se întâmplă cu acesta.

Afișarea / Ascunderea elementelor

Sunt utile proprietățile *display* și *visibility*. Valoarea *none* pentru proprietatea *display* ascunde elementul ca și când acesta ar lipsi din document. *Visibility* poate avea valorile *visible* sau *hidden.* Valoarea *hidden* ascunde conținutul elementului dar păstrează locul în aspectul paginii. Vezi exemplul https://playcode.io/1698908.

Stiluri pentru ancore

Link-urile au aspect diferit atunci când nu au fost vizitate niciodată, când au fost vizitate, când mouse-ul se află deasupra și când se dă click pe el. Aspectul lor, pentru aceste situații, este dat de navigator, dar poate fi modificat cu ajutorul limbajului CSS, folosind pseudo-clasele. Pseudo-clasele (pseudo-classes) sunt utile pentru a stabili caracteristicile unor stări specifice ale unui element. În cazul link-urilor stările sunt: *link* (nevizitat), *visited* (vizitat), *hover* (mouse deasupra) și *active* (click deasupra). Sintaxa este:

```
a:stare {
   proprietate:valoare;
   proprietate:valoare;
   …
}
```

Setul de proprietăți stabilește comportamentul pentru fiecare stare pe care o definim. Pentru o funcționare corectă, definirea trebuie făcută în ordinea de mai sus a proprietăților, chiar dacă unele dintre ele lipsesc. Exemplu (lipsește *a:visited*):

```
a:link {
   color:gray;
   text-decoration:none;
}
a:hover {
   color:blue;
   text-decoration:underline;
}
a:active {
   color:yellow;
   background-color:darkgreen;
}
```

Utilizarea pseudo-claselor nu se limitează la ancore. În exemplul anterior (https://playcode.io/717886/), adăugați următoarea linie de cod în fila *styles.css*:

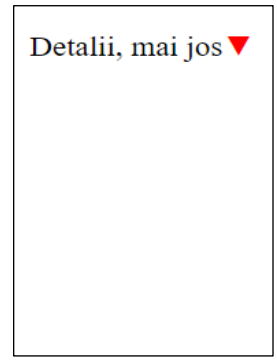
```
#d1:hover {cursor:pointer}
```

apoi treceți cu mouse-ul peste div-ul *d1* și observați cum se modifică aspectul cursorului.

În multe cazuri este nevoie să adăugăm conținut la începutul sau după un anumit element. Pentru asemenea situații este util pseudo-selectorul **::before**, respectiv **::after**.

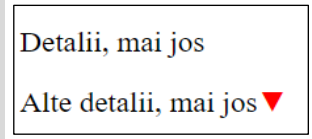
Exemplu:

```
<!DOCTYPE html>
<HTML>
<HEAD>
      <style>
      p::after {
            content:"▼";
            color:red;
      }
      </style>
</HEAD>
<BODY>
      <p>Detalii, mai jos</p>
</BODY>
</HTML>
```



Dacă se dorește aplicarea doar unor selectori aparținând unei clase:

```
<!DOCTYPE html>
<HTML>
<HEAD>
    <style>
        p.second::after {
            content:"▼";
            color:red;
        }
   </style>
</HEAD>
<BODY>
      <p>Detalii, mai jos</p>
      <p class="second">Alte detalii, mai jos</p>
</BODY>
</HTML>
```



Responsivitate

Prin responsivitate se înțelege acea tehnică de proiectare a paginii web care sa îi asigure capacitatea de a se adapta la dimensiunile *viewport*-ului, astfel încât să permită o redare bună a conținutului pe o multitudine de dispozitive: desktop cu ecran mare sau mediu, tabletă sau telefon mobil.

Prin *viewport* se înțelege zona vizibilă utilizatorului dintr-o pagină Web. Pe unele dispozitive, cum ar fi PC-urile, aceasta poate fi redimensionată, modificând astfel

dimensiunea viewport-ului. Pe alte dispozitive, cum ar fi telefoanele smart, *viewport*-ul are dimensiune fixă, în dependentă cu rezoluția ecranului.

Odată cu răspândirea accesului mobil la site-urile Web acestea au simțit nevoia să pună la dispoziția utilizatorilor versiuni adaptate dispozitivelor mobile, care au viewport-uri de dimensiuni mai mici. În acest scop există două posibilități:

- se concepe o versiune a site-ului cu totul nouă, diferită, pentru dispozitive mobile, situație în care serverul detectează platforma de pe care clientul accesează site-ul și îl dirijează spre versiunea corespunzătoare, după caz (vezi site-ul https://hotnews.ro și, respectiv, https://m.hotnews.ro).
- se stabilesc definiți CSS diferite pentru elementele din pagini, bazat pe capacitatea navigatoarelor moderne de a detecta dimensiunea *viewport*-ului.

Prima soluție are dezavantajul că aplicațiile (programele care controlează conținutul paginilor) trebuie rescrise sau reutilizate pentru noua versiune, ceea ce complică mult atât proiectarea cât și mentenanța. Mai mult, dispozitivele mobile au ecrane de dimensiuni variabile astfel încât chiar și varianta dedicată lor poate avea un aspect neatrăgător pe unele dintre ele (de exemplu pe tablete, dacă site-ul a fost proiectat pentru telefoane mobile).

A doua soluție are avantajul că aplicațiile rămân nemodificate, doar elementele se afișează diferit de la un dispozitiv de vizualizare la altul. În plus, această abordare permite stabilirea de definiții diferite pentru o multitudine de tipuri de dispozitive mobile, cu ferestre de dimensiuni într-o plajă foarte largă.

Stabilirea viewport-ului

Pentru a beneficia de această ultimă posibilitate trebuie utilizat un tag suplimentar, obligatoriu, în antetul documentului:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Acest tag „instruiește” navigatorul să urmărească lățimea ferestrei prin care utilizatorul privește pagina (viewport) și să aplice un zoom inițial de 100% pe conținutul paginii, relativ la lățimea ferestrei de vizualizare.

Interogări media

Stabilirea de stiluri diferite pentru elemente, funcție de lățimea ferestrei, de utilizează directive CSS (query) de forma:

```
@media not|only mediatype and (expressions) {
```

```
CSS-Code;
}
```

Atributele *not* sau *only* stabilesc modul în care definițiile CSS care urmează se aplică sau nu mediului (mediatype) în condițiile îndeplinirii condițiilor din expresia *expressions*. Mediatype poate fi ecranul (screen), imprimanta (print), sinteză vocală (aural) etc.

Problema poate fi abordată în două moduri simetrice: A) întâi se proiectează aspectul pentru ecrane mari, după care se utilizează interogările @media pentru a stabili aspectul pentru viewport-uri mai mici sau B) se proiectează mai întâi aspectul pentru cel mai mic viewport după care se utilizează interogările @media pentru a stabili aspectul pentru dispozitive cu ecran mai mare. Este recomandată metoda B).

Exemplul de la adresa https://playcode.io/ precum și dimensiunea fontului titlului H1, la pragul de 480px[12] lățime a viewport-ului.

Codul html este:

```
<head>
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
</head>
<body>
  <header><h1>Aici antetul paginii</h1></header>
  <main></main>
  <footer></footer>
<body>
```

iar codul CSS este:

```
body {
  background: white;
  font-family: Helvetica neue
}
header{
  height:100px;
  text-align:center;
  padding-top:50px;
  background-color: coral;
}
h1{font-size: 14pt;}
```

[12] Nu sunt pixelii fizici ci așa-ziși pixeli CSS. Pe aceeași lățime a viewport-ului, aparatele pot avea număr diferit de pixeli fizici: https://viewportsizer.com/devices/

```
@media screen and (min-width:640px){
  h1{
   font-size:18pt;
  }
  header {
   height:200px;
   padding-top:100px;
   background-color: cornflowerblue ;
  }
}
```

Se pot utiliza praguri multiple pentru a optimiza aspectul la mai multe dispozitive. Exemplul de la adresa https://playcode.io/718888/ modifică aceleași caracteristici ca la exemplul anterior, dar la 2 praguri: 360px și 480px[13]. Rezultatul poate fi văzut în Figura 31.

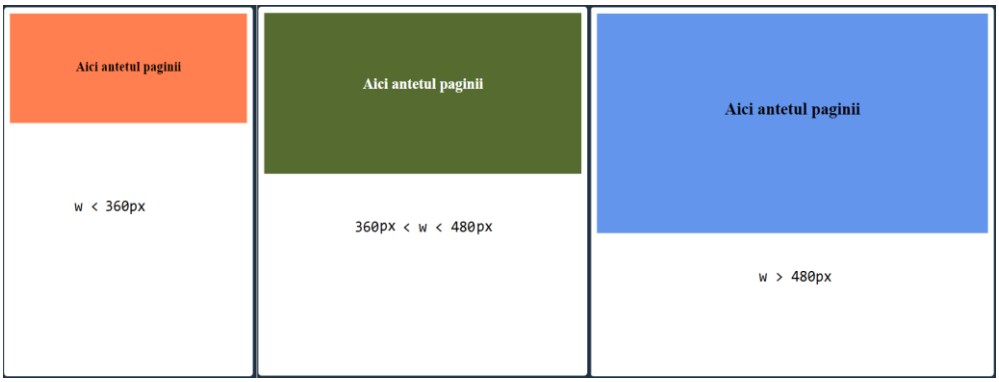


Figura 31. Modificarea proprietăților CSS ale antetului în funcție de lățimea viewport-ului, cu 3 praguri

Ce se modifică, față de exemplul anterior, este doar codul CSS:

```
@media screen and (min-width:360px){
  h1{
   font-size:16pt;
   color:white;
  }
  header {
   height:150px;
   padding-top:70px;
   background-color: darkolivegreen ;
  }
}
@media screen and (min-width:480px){
```

[13] 360 si 480 px sunt cele mai comune lățimi de viewport la majoritatea telefoanelor mobile

```
  h1{
   font-size:18pt;
   color:black;
  }
  header {
   height:200px;
   padding-top:100px;
   background-color: cornflowerblue ;
  }
}
```

Un alt exemplu, https://playcode.io/718901/, modifică poziția a 3 elemente *div*, din secțiunea *main* a documentului, la aceeași două praguri. Rezultatul este similar cu cel din Figura 32.

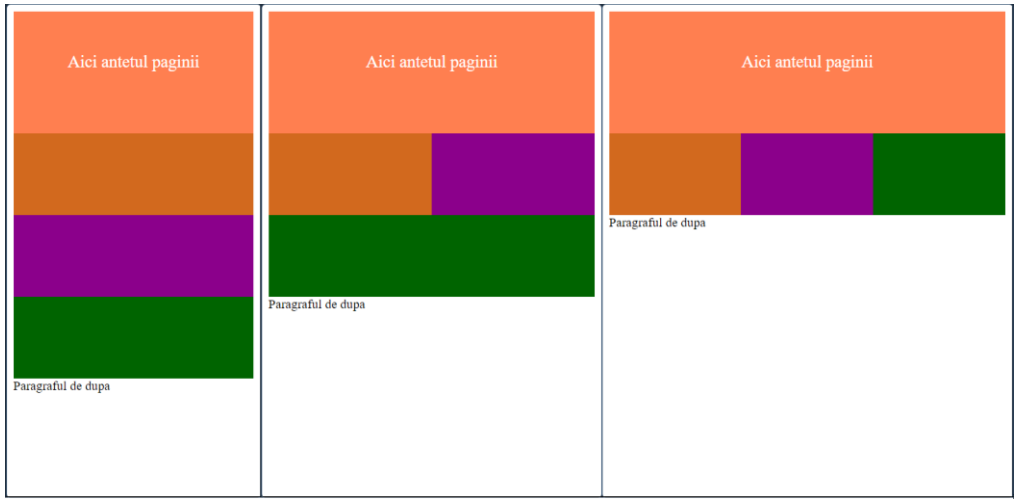


Figura 32. Modificarea layout-ului în funcție de lățimea viewport-ului, cu 3 praguri

La o lățime a *viewport*-ului sub 360 px, cele trei elemente se află unul sub altul, având o lățime de 100% din lățimea *viewport*-ului. La o lățime a *viewport*-ului cuprinsă între 360 px și 480 px, primele două div-uri au o lățime de 50% din cea a *viewport*-ului și sunt afișate unul lângă altul, iar al treilea div are o lățime de 100% din a *viewport*-ului, fiind afișată sub primele două. La o lățime a *viewport*-ului de peste 480 px, cele 3 div-uri ocupă, fiecare, o lățime egală cu 33,3% din cea a *viewport*-ului, plasându-se unul lângă altul.

Codul HTML este redat mai jos:

```
<head>
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
</head>
```

```
<body>
<header>Aici antetul paginii</header>
<main>
 <div class="left" id="d1"></div>
 <div class="left" id="d2"></div>
 <div class="left" id="d3"></div>
 <p>Paragraful de dupa</p>
</main>
<footer></footer>
<body>
```

iar codul CSS corespunzător este următorul:

```
body {background: white;font-family: Helvetica neue}
* {border-size:border-box}
header{
  height:100px;
  text-align:center;
  padding-top:50px;
  background-color: coral;
  font-size:16pt;
  color:white;
}
.left {
  float:left;
  height:100px;
}
p{clear:left}
#d1,#d2,#d3 {width:100%;}
#d1 {background-color:chocolate}
#d2 {background-color:darkmagenta}
#d3 {background-color: darkgreen}
@media screen and (min-width:360px){
  #d1,#d2{
   width:50%
  }
  #d3{
   width:100%;
  }
@media screen and (min-width:480px){
  #d1,#d2,#d3 {width:33.3%;}
}
```

Exemplul care urmează (https://playcode.io/719166/) colapsează bara de navigare (cu link-urile de navigare) într-un buton care simbolizează meniul, atunci când

lățimea *viewport*-ului este mai mică de 480px. Rezultatul se poate vedea în Figura 33.

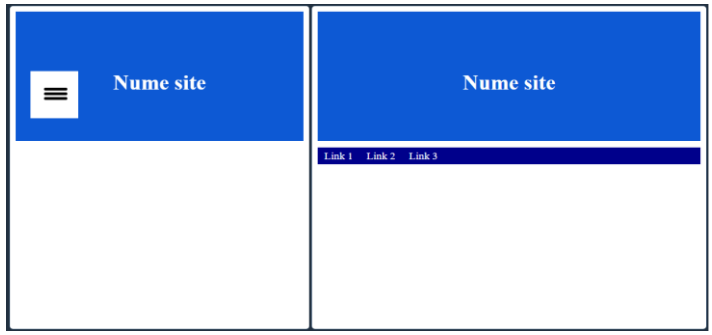


Figura 33. Modificarea aspectului meniului în funcție de lățimea viewport-ului.

Documentul are o structură extrem de simplă, compusă din antet (header), bara de navigare (nav) și secțiunea principală (main):

```
<!DOCTYPE html>
<html>
<head>
<title></title>
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<link rel="stylesheet" href="css_responsive.css">
</head>
<body>
<div id="container">
<header>
  <h1>Nume site</h1>
</header>
<nav><span>Link 1</span><span>Link 2</span><span> Link 3
</span></nav>
<main></main>
</div>
</body>
</html>
```

Pentru obținerea efectului dorit merită studiat puțin conținutul fișierului *css_responsive.css*. Pentru toate dispozitivele, elementele de tip box au proprietatea *box-sizing* setată *border-box* iar corpul documentului are stabilite proprietățile pentru tipul fontului și culoarea fundalului.

```
body {background: white;font-family: Helvetica neue}
* {box-sizing:border-box;}
```

Proiectarea se face, de regulă, „de la mic la mare”, încât pentru ecranele mici, sub 480 px, definițiile elementelor *header* și *nav* sunt cele generale:

```
header {
  height:200px;width:auto;
  background-color:#0d59d4;
  text-align:center;
  color:white;
  padding-top:70px;
}
nav {
  position:absolute;
        height:70px;width:70px;
        top:100px; left:30px;
        color:white;
        background-image:
url('https://cdn4.iconfinder.com/data/icons/flat-
black/512/menu.png');
        background-size:cover;
  background-color: white;
}
nav span {display:none;}
```

Meniul *nav* este constituit dintr-un pătrat cu laturile de 70 px, poziționat *absolut* la 100 px față de marginea de sus și 30 px față de marginea stângă a elementului *container*. Elementul *nav* are o imagine de fundal reprezentand pictograma unui meniu de tip „hamburger”, imagine luată direct de pe Internet. O observație aparte trebuie făcută cu privire la modul cum este redată imaginea de fundal. Dimensiunea originală a imaginii este de 512 / 512 px, adică mult mai mare decât a elementului nav (70 / 70 px). Pentru a face imaginea să încapă și să fie afișată complet în spațiul rezervat, se folosește proprietatea *background-size* a elementului *nav*. Valoarea *cover* face ca imaginea să fie comprimată pentru a se potrivi în elementul nav. Nu întâmplător raportul lățime/ înălțime a elementului nav a fost ales egal cu cel corespunzător al imaginii, adică 1.

Despre comportamentul produs de diferitele valori ale proprietății *background-size* puteți citi aici.

Pentru *viewport*-urile cu lățimea de cel puțin 480 px, definițiile elementului *nav* se modifică:

```
@media screen and (min-width:480px) {
   nav {
     position:relative;
     top:10px; left:0;
     width:100%;height:26px;
```

```
      background-color:darkblue;
      background-image: none;
    }
    nav span {
      display:inline-block;
      padding:4px 10px;
    }
}
```

El devine poziționat relativ, după elementul header, la o distanță de 10 px de acesta. Lățimea se modifică la 100% iar înălțimea la 26px. De asemenea se modifică culoarea de fundal și se elimină imaginea de fundal cu directiva `background-image: none;`. În această situație, devine vizibil conținutul elementului *nav* din fișierul html:

```
<nav><span>Link 1</span><span>Link 2</span><span>Link 3
</span></nav>
```

Acest conținut este stilizat pentru *viewport*-uri mai mari de 480 px:

```
nav span {
      display:inline-block;
      padding:4px 10px;
}
```

și ascuns la dimensiuni mai mici:

```
nav span {display:none;}
```

La aceste tehnici de bază se pot adăuga câteva trucuri. În ceea ce privește imaginile, se vor folosi dimensiuni relative, exprimate în procente. O imagine având proprietatea `width:100%` relativ la lățimea containerului, se va redimensiona în ambele sensuri pentru a ocupa mereu 100% din lățimea containerului. Dacă imaginea originală are o lățime de doar 300px, la un viewport de 1200 px ea se va mări de 4 ori, astfel încât calitatea redării scade foarte mult. De aceea se preferă proprietatea `max-width`, care permite redimensionarea imaginii în ambele sensuri, dar fără a depăși procentul stabilit din dimensiunea imaginii originale.

La adresa https://playcode.io/720656/ aveți aceeași imagine, având lățimea de 320px, așezată una sub alta, în care prima are proprietatea `width:100%` iar a doua `max-width:100%`. Ambele lățimi sunt raportate la cea a containerului, care are lățimea de 80% din cea a ferestrei navigatorului. Redimensionați fereastra cu rezultatul și observați diferența, care este reprodusă și în Figura 34.

În partea stângă, containerul are lățimea mai mare de 320px și prima imagine este scalată peste dimensiunea ei originală, în timp ce a doua rămâne la 320px. În dreapta, lățimea este mai mică de 300px și ambele imagini sunt scalate „în jos”.

Alt truc este folosirea dimensiunilor relative la viewport pentru fonturi. Este utilă unitatea de măsură vw, (viewport width), care reprezintă procent din lățimea *viewport*-ului (1vw=1%).

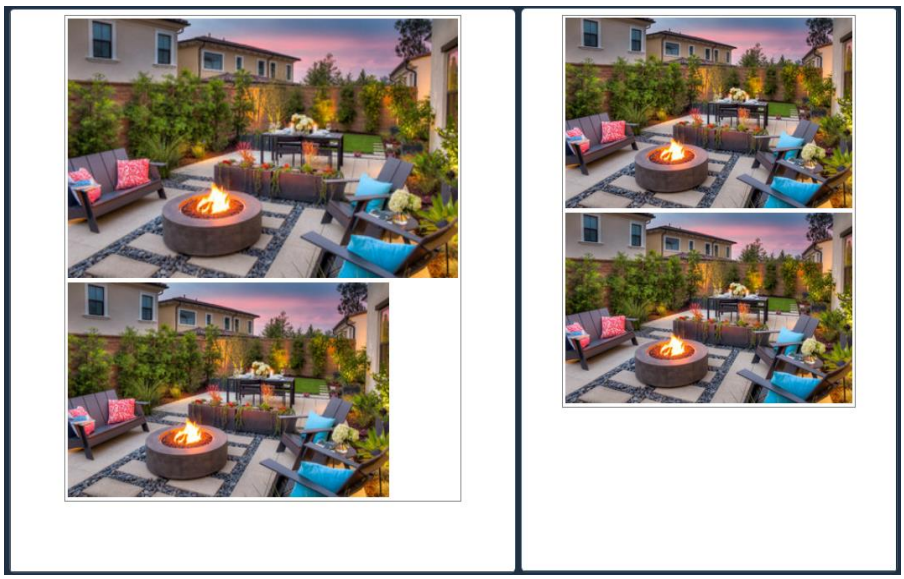


Figura 34. Exemplu de comportament al imaginilor în funcție de modul de stabilire al lățimii lor, relativ la a containerului

Exemplul următor păstrează constantă dimensiunea fontului titlului H1, la valoarea de 14pt, pentru lățimi ale viewport-ului sub 480px și crește proporțional cu aceasta la lățimi mai mari de 480px.

HTML

```
<head>
  <meta name="viewport"
  content="width=device-width, initial-scale:1.0">
</head>
<body>
  <head>
    <h1>Titlul paginii</h1>
  </head>
</body>
```

CSS

```
body {background: white;font-family: Helvetica neue}
h1 {
  font-size:14pt;
}
@media screen and (min-width:480px){
  h1 {
    font-size:4vw;
  }
}
```

Exemplul poate fi văzut la adresa https://playcode.io/761772/

Modelul GRID

Pentru proiectarea responsive a paginilor Web există câteva biblioteci CSS care facilitează crearea ușoară a conținutului. Cele mai populare cinci biblioteci, conform sunt (Bose, Top 5 CSS Frameworks for Developers and Designers, 2023), în ordine: Bootstrap, Tailwind CSS, Foundation, Bulma și Skeleton. Majoritatea bibliotecilor se bazează pe un sistem de tip grid (grilă), care presupune împărțirea viewport-ului în coloane și rânduri, la intersecția lor obținându-se celule care conțin informația de afișat.

Numărul de coloane variază de la o bibliotecă la alta dar în cele ce urmează vom considera un grid cu 6 coloane, pentru a nu complica excesiv explicația dar suficient pentru a surprinde ideea.

Lățimea unei coloane poate varia de la 100% din lățimea ecranului până la 1/6 din lățime, Figura 35.

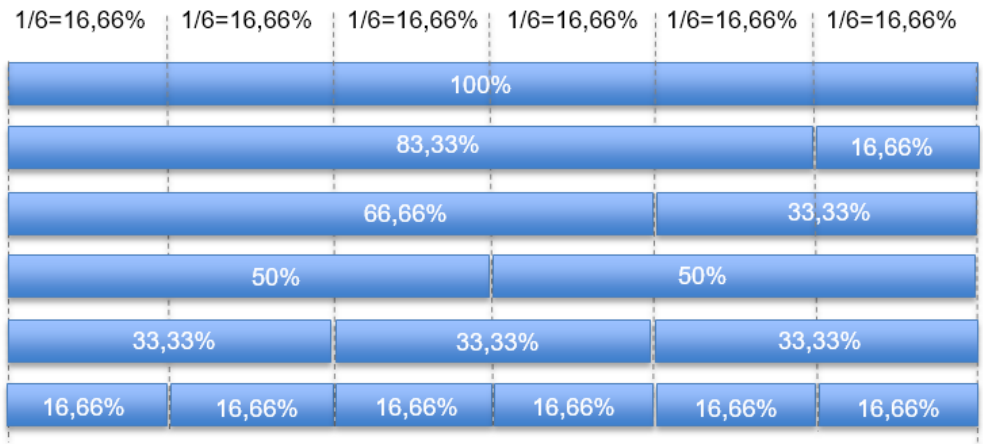


Figura 35. Sistemul grid cu 6 coloane

Pe fiecare rând pot exista un număr variabil de coloane, condiția fiind ca suma lățimilor lor sa fie 100%, Figura 36.

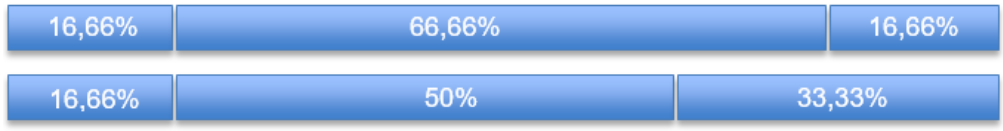


Figura 36. Variantă de layout, cu 2 rânduri și trei coloane cu lățimi diferite

Pentru fiecare lățime de coloană se definește câte o clasă, de forma *.col-n*, în care *n* reprezintă multiplul lățimii minime de 16,66%. Astfel, *.col-1* este coloana cea mai mică, cu lățimea de 16,66%, *.col-2* este coloana cu lățime dublă, 33,33%, *.col-6* este coloana cu lățimea de 6 ori mai mare, adică 100%.

```
.col-1 {width: 16.66%;}
.col-2 {width: 33.33%;}
.col-3 {width: 50%;}
.col-4 {width: 66.66%;}
.col-5 {width: 83.33%;}
.col-6 {width: 100%;}
```

Toate coloanele trebuie să aibă proprietatea *float: left*. Atribuirea proprietăților tuturor claselor de tip .col-* se face cu sintaxa [class*="col-"] {…..}. Vom mai atribui un mic padding și un chenar tuturor coloanelor pentru aspect și evidențiere:

```
[class*="col-"] {
   float: left;
   padding:10px;
   border: 1px solid blue;
}
```

Pentru ca aspectul să fie același în orice navigator, toate elementele vor avea proprietatea *box-sizing: border-box;*

```
*{
  box-sizing: border-box;
}
```

Toate coloanele se plasează în interiorul câte unui rând. Pentru ca rândurile să se așeze unele sub altele, trebuie ca proprietatea *float* a coloanelor să fie anulată. Vom folosi o clasă .row care va fi atribuită fiecărui rând, astfel:

```
.row::after {
  content: "";
  clear: both;
  display:table;
```

Aici, proprietatea *display* are valoarea *table*, o valoare mai puțin utilizată dar utilă în acest caz deoarece face ca fiecare rând să fie afișat ca un tabel[14] .

Puse toate la un loc, rezultă codul:

```
<!DOCTYPE html>
<html>
<head>
     <meta    name="viewport"    content="width=device-width,
initial-scale=1">
    <style>
    * {
        box-sizing: border-box;
    }
    .row::after {
        content: "";
        clear: both;
        display: table;
    }
    .col-1 {width: 16.66%;}
    .col-2 {width: 33.33%;}
    .col-3 {width: 50%;}
    .col-4 {width: 66.66%;}
    .col-5 {width: 83.33%;}
    .col-6 {width: 100%;}

    [class*="col-"] {
        float: left;
        padding:10px;
        border: 1px solid blue;
    }
    </style>
</head>
<body>
    <div class="row">
        <div class="col-2">Col 1</div>
        <div class="col-3">Col 2</div>
        <div class="col-1">Col 3</div>
    </div>
</body>
</html>
```

[14] Alte valori posibile sunt *table-row* și *table-cell*. Exemple de utilizare a acestor valori găsiți la https://www.freecodecamp.org/news/the-css-display-property-display-none-display-table-inline-block-and-more/

În acest moment pagina nu arată bine pe telefonul mobil, Figura 37. Tot ceea ce face codul este să afișeze coloanele proporțional cu lățimea ecranului.

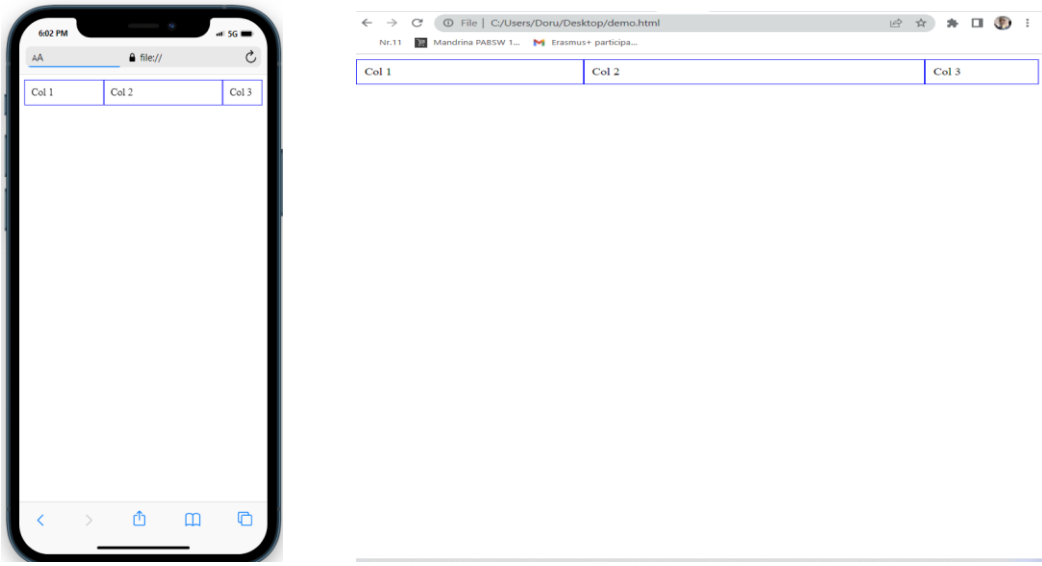
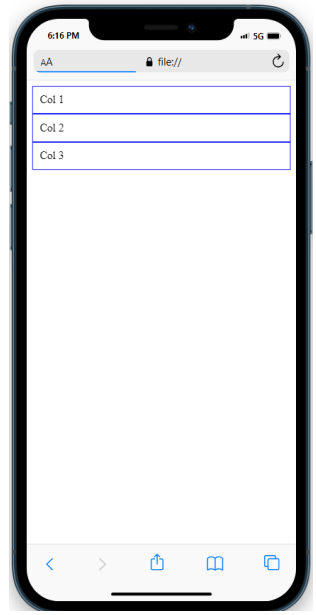


Figura 37. Pagina corespunzătoare codului de mai sus, pe smartphone și pe desktop

Pentru a face pagina cu adevărat responsive, apelăm la „media query”:

```
@media only screen and (max-width: 768px) {
    /* pentru ecrane mici: */
      [class*="col-"] {
        width: 100%;
      }
}
```

După aceasta, pe telefon pagina va arăta ca în imaginea alăturată.



Un exemplu complet, cu trei rânduri cu lățimi diferite puteți vedea la adresa:

https://playcode.io/1529366

În mod clar organizarea pe 6 coloane este insuficientă, pagina nu arată prea bine pentru lățimi ale viewport-ului între 600px și 1280px, coloanele laterale fiind prea înguste comparativ cu coloana centrală.

Uzual, bibliotecile CSS folosesc modelul grid cu 12 coloane.

Sistemul flexbox

Sistemul flexbox (Flexible Box Layout), facilitează proiectarea unei structuri flexibile de aspect responsive, fără a utiliza proprietatea *float* sau poziționarea. Sistemul este cunoscut și sub numele de *modul*, deoarece flexibilitatea nu este o simplă proprietate ci constă într-un ansamblu de proprietăți aplicabile atât containerului cât și elementelor flexibile conținute de acesta.

Astfel, prima proprietate a unui element de tip container (zis și părinte), care va conține elemente flexibile - copiii, este *display* cu valoarea *flex*:

```
.container {
   display: flex;
}
```

Elementele flexibile pot fi ordonate pe rând sau pe coloană, de la stânga la dreapta sau de la dreapta la stânga, respectiv de sus în jos și de jos în sus, Figura 38.

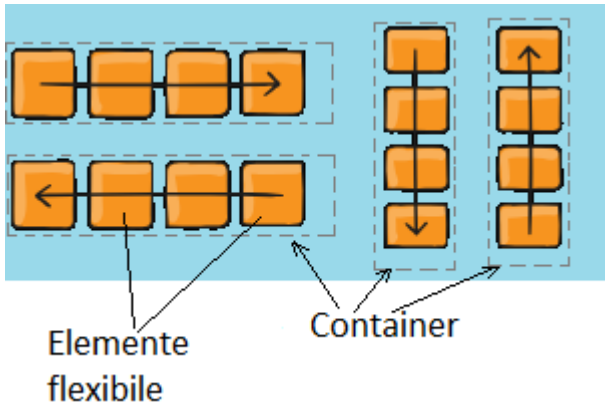


Figura 38. Ordonarea elementelor flexibile în interiorul containerului

Modul de ordonare a elementelor flexibile se stabilește cu proprietatea *flex-direction*, care poate avea una din valorile: *row*, *row-reverse*, *column*, *column-reverse*.

```
.container {
   display: flex;
   flex-direction: row; /*sau row-reverse etc */
}
```

O altă proprietate a containerului este *flex-wrap*, care stabilește cum se înfășoară elementele flexibile. Implicit ele vor încerca să stea toate pe un rând, echivalent cu valoarea *nowrap* a proprietății. Celelalte două valori admise sunt *wrap* și *wrap-reverse*.

```
.container {
   display: flex;
   flex-direction: row; /* sau row-reverse etc */
   flex-wrap: wrap /* sau wrap-reverse */
}
```

O altă proprietate specifică, care stabilește distribuția pe direcția specificată de *flex-direction* este *justify-content* care poate avea una din valorile*: flex-start, flex-end, center, space-between, space-around, space-evenly*. *Flex-start* și *flex-end* aliniază elementele flexibile pornind de la începutul, respectiv sfârșitul containerului, fără spațiu între ele. *Center* aliniază elementele pornind de la centru spre margini. *Space-between* aranjează elementele uniform pe orizontală, primul element fiind aliniat cu începutul containerului iar ultimul cu sfârșitul containerului. *Space-around* aliniază elementele cu spații egale între ele, dar față de începutul și sfârșitul containerului spațiile sunt reduse la jumătate. În fine, *space-evenly* face spațiile egale, inclusiv față de marginile containerului.

În ceea ce privește alinierea pe cealaltă axă (verticală, în cazul în care *flex-direction* are valoarea *row*), proprietatea este *align-items* cu una din valorile: *stretch* (implicit), *flex-start*, *flex-end*, *center* și *baseline*. Exemple în Figura 39.

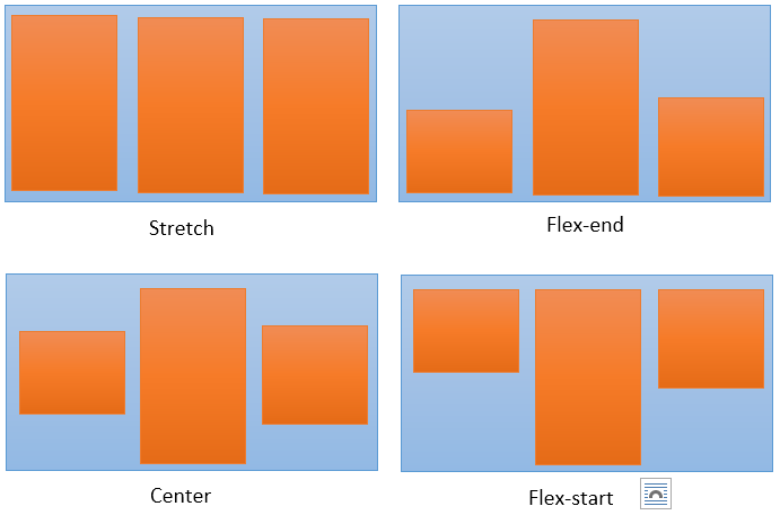


Figura 39. Exemple de aliniere cu align-items

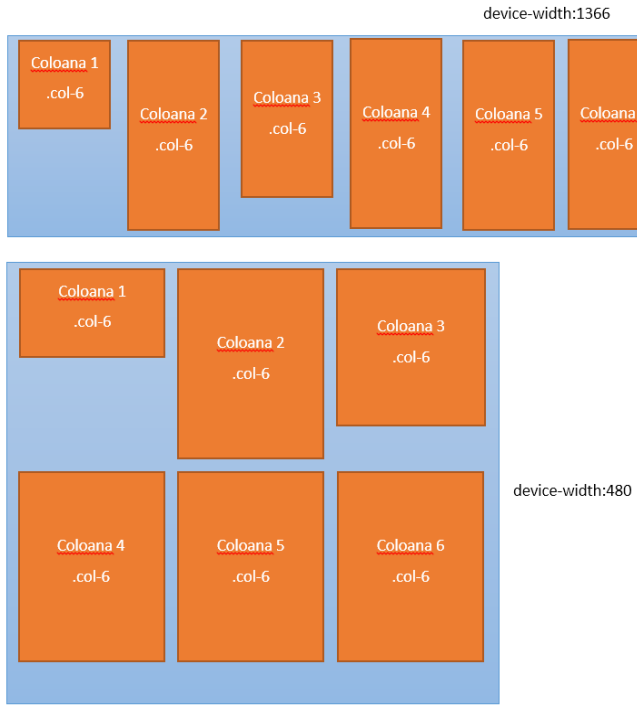


Figura 40. Layout obținut cu sistemul GRID la două dimensiuni diferite ale dispozitivului

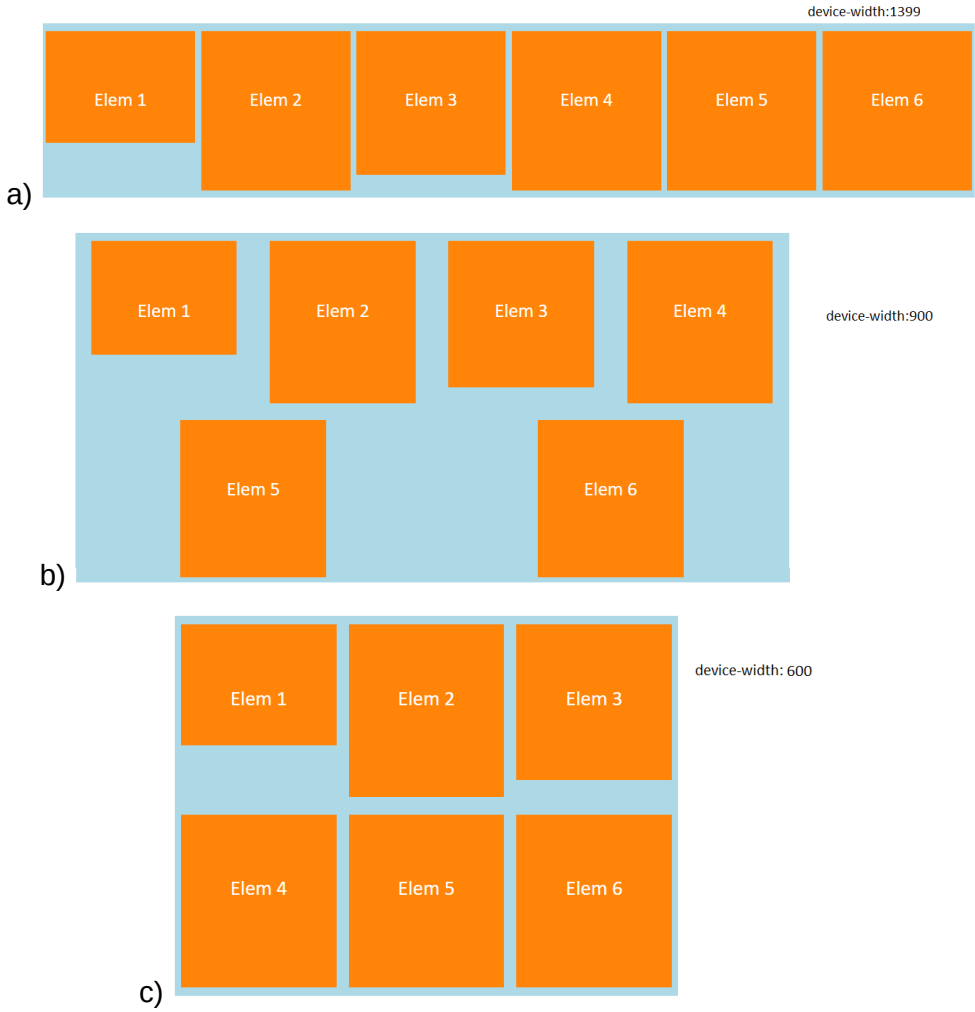


Figura 41. a) Layout-ul din figura 38, obținut cu modulul flex, la viewport de 1399 px. b) Layout-ul pentru viewport de 900px c) Layout-ul pentru viewport de 600px

Exemplul din Figura 41 poate fi văzut la adresa https://playcode.io/1584344.

Figura 42 redă un layout destul de comun pentru multe pagini Web. El este realizat integral folosind modulul *flex*, în combinație cu interogări *@media query*. Punctul de inflexiune este stabilit la 660px. Întregul exemplu poate fi accesat la adresa https://playcode.io/1584607.

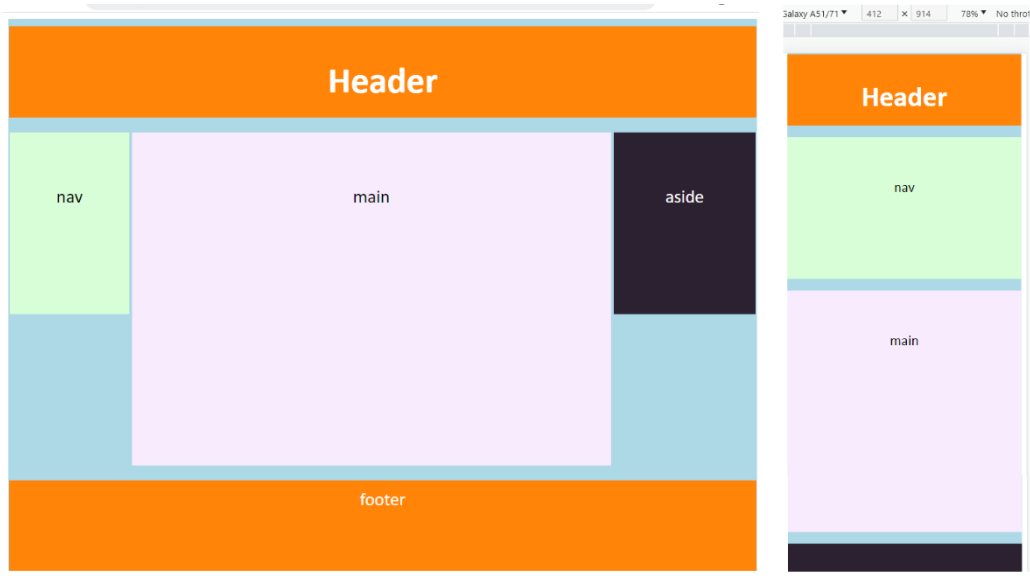


Figura 42. Layout responsive realizat cu modulul flex

Un tutorial instructiv despre crearea de layout-uri pentru pagini Web folosind modulul flexbox este la adresa https://coder-coder.com/build-flexbox-website-layout/.

Bootstrap

Bootstrap este una din cele mai populare biblioteci CSS utilă pentru crearea site-urilor Web de la 0. După athemes.com (Pattakos, 2023), primele cinci locuri din topul celor nouă cele mai bune biblioteci CSS sunt ocupate de:

1. Bootstrap
2. Foundation
3. Tailwind CSS
4. Bulma
5. Ulkit

Se poate constata că Bootstrap ocupă primul loc atât în clasamentul athemes cât și browsertack. De aceea ne vom ocupa de ea în următoarele pagini.

Bootstrap este un un set de instrumente (toolkit) pentru realizarea rapidă de interfețe Web responsive. Versiunea curentă este 5, dar versiunile 3 și 4 sunt încă larg răspândite datorită popularității de care s-au bucurat încă de la început. Versiunea 4 este un upgrade al versiunii 3, în timp ce versiunea 5 diferă prin schimbarea tehnologiei care stă în spatele instrumentului (JavaScript în loc de jQuery).

Sunt două posibilități de utilizare:

1. se descărcă fișierul arhivă .zip de la sursă (https://getbootstrap.com/docs/4.6/getting-started/download/), se dezarhivează și se plasează într-un subdirector în directorul proiectului,
2. se poate face o legătură cu fișierul stocat pe unul din serverele distribuitoare de conținut (CDN - Content Delivery Network).

În primul caz, să presupunem că am plasat fișierul dezarhivat *bootstrap.css* în directorul css.

Link-ul către acest fișier se plasează în antetul fiecărei pagini în forma:

```
<link rel="stylesheet" href="css/bootstrap.css">
```

Arhiva conține mai multe fișiere css, din care unul este varianta comprimată a fișierului *botstrap.css*, numit *bootstrap.min.css*. Acesta este mai mic cu câțiva KB față de versiunea necomprimată, prin urmare se încarcă ceva mai repede în browser. Legătura va fi făcută, după caz, astfel:

```
<link rel="stylesheet" href="css/bootstrap.min.css">
```

Pentru funcționare se dezarhivează și fișierul *bootstrap.js* și se plasează într-un subdirector, de exemplu *scripts*. Integrarea fișierului cu scripturi în pagini se face inserând următoarele linii în antetul fiecărui document:

```
<script src="scripts/bootstrap.js">
</script>
```

Și acest fișier are o variantă comprimată, *bootstrap.min.js*, caz în care legătura se face cu liniile de cod:

```
<script src="scripts/bootstrap.min.js">
</script>
```

În al doilea caz, link-ul se face către fișierele *css* și *js* aflate pe niște servere cu funcție de livrare de conținut, CDN.

```
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/
bootstrap.min.css">
```

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bo
otstrap.min.css">
```

Fiind bazat pe jQuery, Bootstrap 4 are nevoie și de o bibliotecă jQuery, care se inserează și ea în header:

```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jqu
ery.min.js"></script>
```

sau

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jque
ry.min.js"></script>
```

Adesea, pentru obținerea unor efecte speciale, cum ar fi animația, Bootstrap se folosește în conjuncție cu o altă bibliotecă JavaScript, fără ca utilizarea să presupună cunoștințe de JavaScript. Și această bibliotecă se poate apela în header-ul documentului:

```
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.2/dist/js/bo
otstrap.min.js"></script>
```

sau

```
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/boot
strap.min.js"></script>
```

Avantajul utilizării serverelor DNS este acela că foarte multe pagini Web din Internet le folosesc și, odată vizitată o asemenea pagină, biblioteca *css* ca și fișierul *js* rămân memorate în memoria cache a navigatorului. Asta face ca, atunci când un vizitator vă accesează pagina, cele două fișiere sunt accesibile imediat, incomparabil mai repede decât dacă ar fi citite de pe serverul unde găzduiți paginile. Rezultatul este încărcarea mult mai rapidă a paginilor la vizitator.

Bootstrap constă într-o colecție vastă de clase predefinite, care ajută la stilizarea foarte ușoară a elementelor unei pagini și într-o grilă care împarte viewport-ul în coloane care permit personalizarea unui răspuns responsive pentru o gamă largă de dispozitive.

Sunt trei clase înrudite: *container*, *container-fluid* și *container-{breakpoint}*. Acestea se folosesc pentru a încadra conținutul întregii pagini. Prima produce un *div* cu lățime fixă dar dependentă de mărimea viewport-ului, în timp ce a doua produce un div cu lățime egală cu cea a view-portului. A treia clasă, adăugată le versiunea 4, container-{breakpoint}, are lățimea: 100% până la punctul de întrerupere specificat. Punctele de întrerupere sunt predefinite: sm, md, lg și xl (sm - ecrane small, md - ecrane medii, lg - ecrane late, xl - ecrane extra late). Comportamentul este sintetizat în tabelul 1.

Tabel 2 Până la punctul de întrerupere, toate containerele au lățimea de 100%

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
.container	100%	540px	720px	960px	1140px
.container-sm	100%	540px	720px	960px	1140px
.container-md	100%	100%	720px	960px	1140px
.container-lg	100%	100%	100%	960px	1140px
.container-xl	100%	100%	100%	100%	1140px
.container-fluid	100%	100%	100%	100%	100%

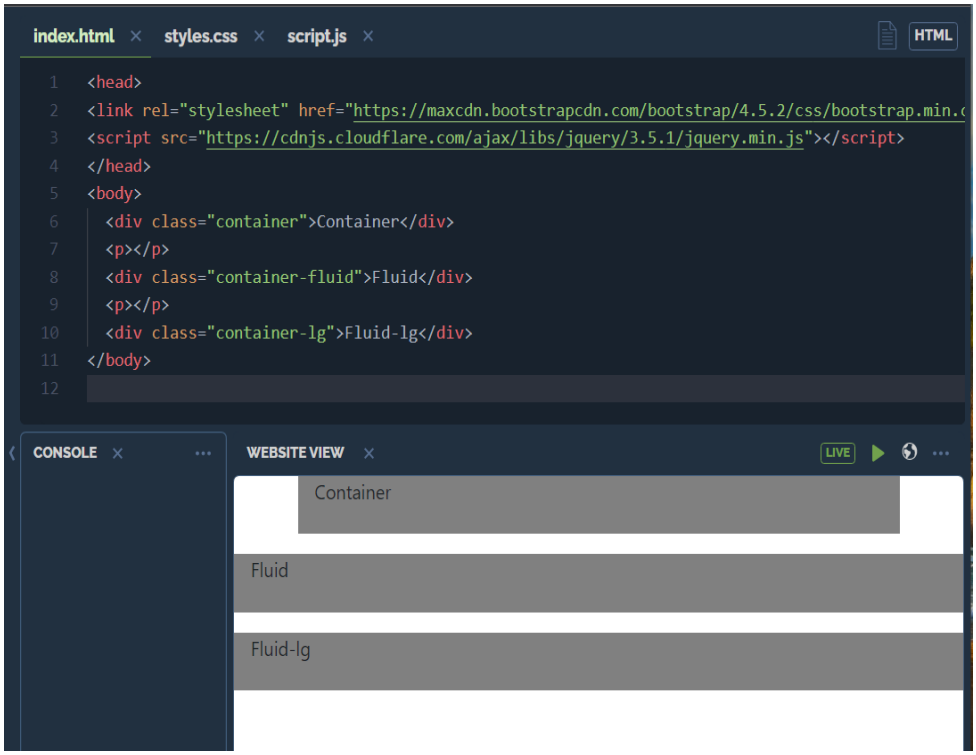


Figura 43. Ilustrarea exemplului de mai sus. Glisati linia de demarcație dintre cele două cadre pentru a observa comportamentul diferit al celor două containere

Un exemplu de utilizare a clasei container este la adresa https://playcode.io/1542649. Analizați ambele file: index.html și styles.css.

Trageți stânga-dreapta de linia de demarcație a cadrelor *Console* și *Website View* pentru a observa comportamentul.

Observație: culoarea gri containerului a fost adăugată în fila *styles.css* pentru a pune în evidență limitele containerului.

Asemănătoare clasei *container* este clasa *jumbotron* cu varianta *jumbotron-fluid*. Acestea două din urmă sunt folosite pentru a scoate în evidență o secțiune a documentului care prezintă o importanță deosebită.

Un exemplu poate fi vizualizat la adresa https://playcode.io/846105/.

Sistemul de grilă (grid)

Bootstrap împarte viewport-ul în 12 coloane egale. Este permisă gruparea coloanelor pentru a crea coloane mai late. Sistemul grilă este responsive, coloanele rearanjându-se în funcție de lățimea ecranului. Figura 44. Sistemul grilă. Coloane individuale (sus) și diverse grupări (jos)ilustrează acest sistem, prin reproducerea celor 12 coloane individuale, sus, și a câtorva exemple de grupări, jos.

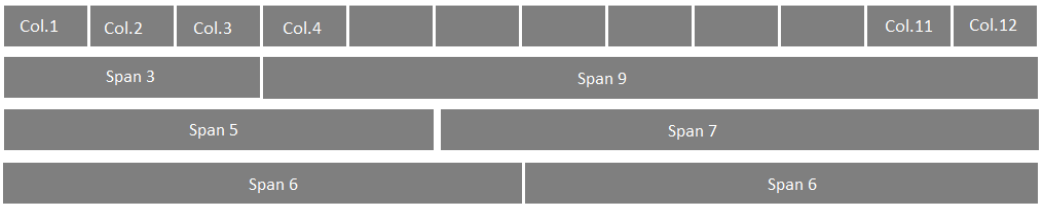


Figura 44. Sistemul grilă. Coloane individuale (sus) și diverse grupări (jos)

Clasele pentru grilă

Sistemul de grilă Bootstrap 4 are cinci clase:

.col- (dispozitive foarte mici - lățimea ecranului mai mică de 576 px)

.col-sm- (dispozitive mici - lățimea ecranului egală sau mai mare de 576 px)

.col-md- (dispozitive medii - lățimea ecranului egală sau mai mare de 768 px)

.col-lg- (dispozitive mari - lățimea ecranului egală sau mai mare de 992 px)

.col-xl- (dispozitive xlarge - lățimea ecranului egală sau mai mare de 1200 px)

Clasele de mai sus pot fi combinate pentru a crea aspecte mai dinamice și mai flexibile[15].

Sistemul este destul de complex și nu poate fi prezentat în întregime în materialul de față.

[15] https://getbootstrap.com/docs/4.6/layout/grid/

Totuși vom da câteva explicații de bază și exemple care să ajute la parcurgerea documentației prin studiu individual. Un rând este un div care aparține clasei row:

```
<div class=”row”>...</div>
```

Clasa *col* împarte ecranul într-un număr de coloane de lățime egală: 2, 3, 4, 6 sau 12, care nu sunt responsive. Exemplul de mai jos împarte ecranul în două coloane egale:

```
<div class="row">
   <div class="col">
     1 of 2
   </div>
   <div class="col">
     2 of 2
   </div>
</div>
```

Pentru coloane cu lățimi variabile variabile, de asemenea non responsive, se folosește clasa *col-n*. Exemplul de mai jos împarte ecranul în 3 coloane, cu lățimile de 3/12, 5/12 și 4/12 din lățimea ecranului.

```
<div class="row">
   <div class="col-3">
     1 din 3
   </div>
   <div class="col-5">
     2 din 3
   </div>
   <div class="col-4">
     3 din 3
   </div>
</div>
```

La adresa https://playcode.io/1539678 găsiți un exemplu cu cele două clase[16].

Pentru un design responsive se folosesc clasele *col-*-**, cum ar fi *col-md-4*, *col-lg-5* etc.

Un exemplu de utilizare se găsește la adresa https://playcode.io/1542665.

Un exemplu complet de realizare a unei pagini responsive folosind sistemul grid este prezentat la adresa https://playcode.io/1539676. Dacă pagina nu este afișată

[16] În exemplele care ilustrează sistemul grid au fost introduse, prin stilizare, chenare albe, pentru a scoate în evidență coloanele. În realitate nu există spații între coloane și nici între rânduri.

de la început, apăsați butonul de deasupra cadrului *view*. Modificați lățimea ferestrei *Website View* pentru a vedea răspunsul paginii!

Tipografia

În contextul Web, tipografia se referă la modul în care sunt redate caracterele (typo) și, mai general, textele, pe o pagină. Bootstrap utilizează o stivă nativă de fonturi care selectează cea mai bună familie de fonturi pentru fiecare sistem de operare și dispozitiv. Are prestabilite dimensiuni și grosimi de font pentru titluri h1…h6, pentru tag-urile strong, em, mark, u etc. Dispune și de clase speciale dacă nu se dorește folosirea tag-urilor amintite. Spre exemplu:

`<p class="h1">h1. Bootstrap heading</p>` este echivalent cu `<h1>h1. Bootstrap heading</h1>`.

Documentația completă o puteți consulta pe site-ul oficial al Bootstrap, la adresa https://getbootstrap.com/docs/4.6/content/typography/

Schimbarea alinierii

Pentru schimbarea alinierii textului, transformarea lui sunt disponibile clasele grupate în *text-utilities* și *color-utilities*.

Astfel, pentru aliniere se folosesc clasele *text-left*, *text-right*, *text-center* și *text-justify*.

Pentru controlul înfășurării textului se folosesc clasele *text-wrap* (înfășurarea este implicită) și respectiv *text-nowrap*. Folosirea acestor clase impune ca elementele să fie de tip *block* sau *inline-block*.

Transformarea textului

Se folosesc clasele *text-lowercase*, *text-uppercase* și *text-capitalize*.

`<p class="text-lowercase">Lorem Ipsum</p>` este echivalent cu

`<p style="text-transform:lowercase">Lorem Ipsum</p>`

Un exemplu de transformări ale textului găsiți la adresa https://playcode.io/1539514.

Fonturi

Pentru grosimea fontului se folosesc clasele *font-weight-bold*, *font-weight-bolder* ((relativ la elementul părinte), *font-weight-normal*, *font-weight-light* și *font-weight-lighter* (relativ la elementul părinte).

Pentru scrierea înclinată există clasa *font-italic*.

Pentru fonturi monospațiate există clasa *text-monospace*.

Un exemplu poate fi văzut la adresa https://playcode.io/1542721.

Culori

Pentru fonturi și fundaluri, Bootstrap oferă câteva clase a căror nume sugerează semnificația fiecăreia (culori semantice). Clasele sunt de forma *text-**, unde * poate fi unul dintre cuvintele *primary*, *secondary*, *success* etc.

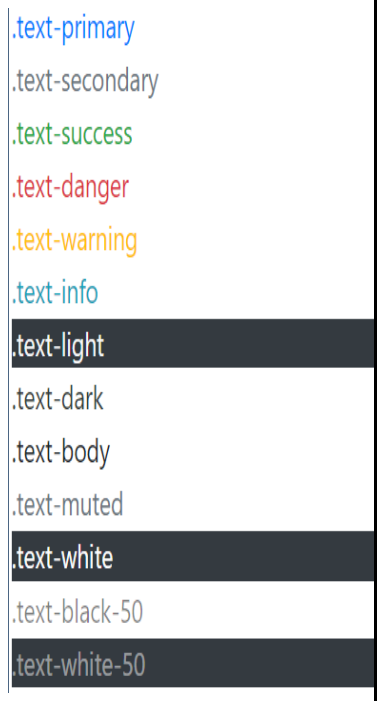
În Tabel 3 sunt trecute în revistă clasele pentru culoarea textului și reproduce rezultatul în coloana a doua. Notați că clasele text-black-50 și text-white-50 sunt redate la fel cu clasa text-muted.

Pentru fundaluri avem clasa *bg-**, unde * are aceleași valori ca la clasa de culori *text-**. În Tabel 4 se redau aceste clase, cu observația că, pentru a asigura o oarecare margine între div-uri, tag-ul *div* a fost stilizat după cum urmează:

```
div {margin:2px; padding:0.1em}
```

Tabel 3 Clase de culori pentru text

```
<p class="text-primary">.text-primary</p>
<p class="text-secondary">.text-secondary</p>
<p class="text-success">.text-success</p>
<p class="text-danger">.text-danger</p>
<p class="text-warning">.text-warning</p>
<p class="text-info">.text-info</p>
<p class="text-light bg-dark">.text-light</p>
<p class="text-dark">.text-dark</p>
<p class="text-body">.text-body</p>
<p class="text-muted">.text-muted</p>
<p class="text-white bg-dark">.text-white</p>
<p class="text-black-50">.text-black-50</p>
<p class="text-white-50 bg-dark"> .text-white-50
</p>
```



Tabel 4 Clase pentru background

.bg-primary .bg-secondary .bg-success .bg-danger .bg-warning .bg-info .bg-light .bg-dark .bg-white .bg-transparent	

Chenare

Clasele pentru chenare permit aplicarea de chenare pe toate laturile sau individual, pe câte una.

Culoarea implicită a chenarului este gri deschis.

La adresa https://playcode.io/1539517 aveți un exemplu de aplicare a chenarelor unui element de tip *span* care nu are chenar implicit (metoda „aditivă”) și de retragere a chenarelor unui element de tip *div* care are, inițial, chenare pe toate laturile. Analizați și fila *styles.css*!

Culoarea chenarelor se stabilește cu ajutorul claselor *border* și *border-**, unde * reprezintă culorile semantice.

Exemplu:

```
<span class="border border-primary"></span>
<span class="border-top-0"></span>
<span class="border-0"></span>
```

Chenarele pot fi rotunjite cu ajutorul claselor *rounded* și *rounded-side*, unde *side* ia valorile *top*, *left*, *bottom* și *right, circle* și *pill*.

Clasa *rounded-pill* produce efect identic cu clasa *rounded-circle* dacă elementul are lățimea și înălțimea egale.

Mărimea rotunjirii se face cu clasele *rounded-sm* și *rounded-lg*. Eliminarea rotunjirilor se face cu clasa *rounded-0*. Vedeți exemplul la adresa https://playcode.io/1538907

Alt exemplu:

```
<p>Border rounded</p>
<span class="border rounded">All</span>
<span class="border rounded-top">Top</span>
<span class="border rounded-left">Left</span>
<span class="border rounded-left rounded-lg">Left-large</span><br>
<span class="border rounded-circle">Circle</span>
<span style="width:160px" class="border rounded-pill">Pill</span>
<div class="border rounded-0">Removed</span>
```

Rezultatul poate fi văzut în Figura 45.

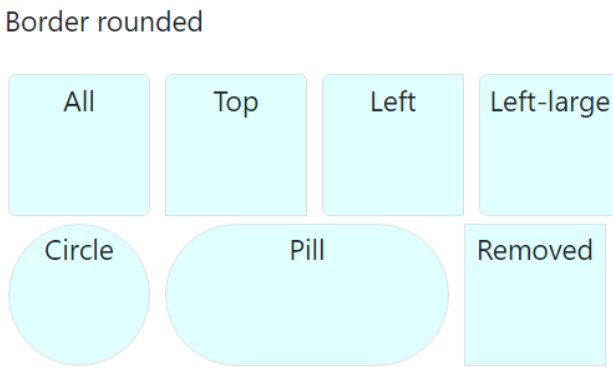


Figura 45. Diverse tipuri de chenare. Elementul cu clasa rounded-pill are lățimea dublă față de înălțime

Culoarea chenarelor se stabilește folosind clasele *border* și *border-**, unde * reprezintă una din culorile semantice: `<span class="border border-success"></span>`

Din păcate, grosimea chenarului nu poate fi stabilită cu Bootstrap 4 (abia cu versiunea Bootstrap 5 este posibil).

Flotabilitatea

Flotabilitatea elementelor se stabilește cu clasa *float-side*, unde *side* are valorile *left*, *right* sau *none*. Pentru responsivitate se specifică și punctul de întrerupere: *float-*-side*, unde * are valorile *sm*, *md*, *lg*, *xlg*. Un exemplu poate fi văzut la adresa https://playcode.io/1537866.

Clasa *clearfix* anulează flotabilitatea elementelor conținute în elementul părinte, având ca rezultat încadrarea tuturor acestor elemente în elementul părinte. Un exemplu este la adresa https://playcode.io/1539784 iar rezultatul este reprodus în Figura 46.

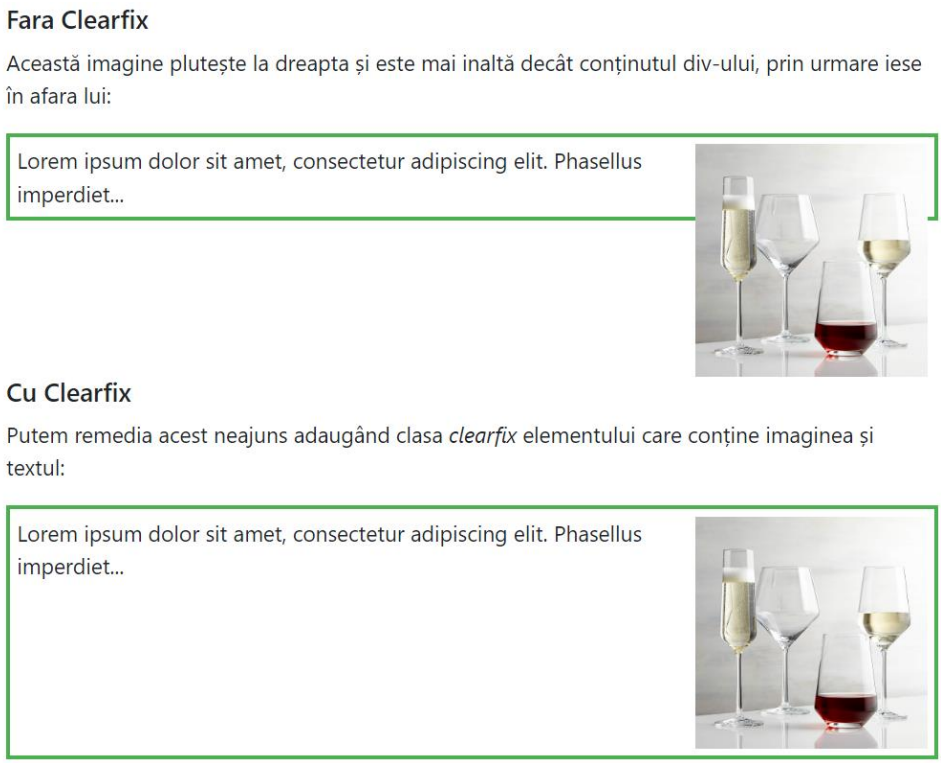


Figura 46. Utilizarea clasei clearfix

Mai există clasa *float-none*, care determină ca elementul cu această clasă să nu plutească la nicio dimensiune a viewport-ului. A se vedea exemplul de la adresa https://playcode.io/1542744.

Butoane

Butoanele dispun de clasele *btn* și *btn-**, unde * este una din culorile semantice. Suplimentar, admite și valoarea *link* pentru link-uri:

```
<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-secondary">Secondary</button>
<button type="button" class="btn btn-success">Success</button>
<button type="button" class="btn btn-danger">Danger</button>
<button type="button" class="btn btn-warning">Warning</button>
<button type="button" class="btn btn-info">Info</button>
<button type="button" class="btn btn-light">Light</button>
<button type="button" class="btn btn-dark">Dark</button>
<button type="button" class="btn btn-link">Link</button>
```

Pentru butoanele care nu au culoare de fundal ci doar contur, este utilă clasa *btn-outline-**, unde * reprezintă una din culorile semantice.

Dimensiunea butoanelor este large(lg) sau small (sm). Clasele corespunzătoare sunt *btn-lg* și *btn-sm*.

Exemplu:

```
<button type="button" class="btn btn-outline-primary">Primary</button>
<button type="button" class="btn btn-outline-dark btn-lg">Dark</button>
<button type="button" class="btn btn-warning btn-sm">Warning</button>
```

Rezultatul este cel din figură:

Imagini

Imaginile folosesc clase ale altor elemente, inclusiv din cele prezentate mai sus, pentru stilizare: *border*, *rounded*, *float*. Există și două clase specifice, *img-fluid*, folosită pentru responsivitate și *img-thumbnail* pentru reprezentarea imaginilor sub formă de miniaturi.

Exemplu:

```
<img src="https://bit.ly/3eTA5H1" class="rounded-lg float-left">

<img src="https://bit.ly/3eTA5H1" class="rounded-lg img-thumbnail">

<img src="https://bit.ly/3eTA5H1" class="rounded-circle img-thumbnail">
```

Rezultatul este ca în Figura 47:

Figura 47. Aspectul imaginilor pentru diverse clase

Legat de imagini, există și tag-ul *figure*, pentru care sunt disponibile clasele *figure*, *figure-img* și *figure-caption*. În combinație cu ele pot fi utilizate aceleași clase ca la imagini.

Un exemplu este disponibil la adresa https://playcode.io/1584621.

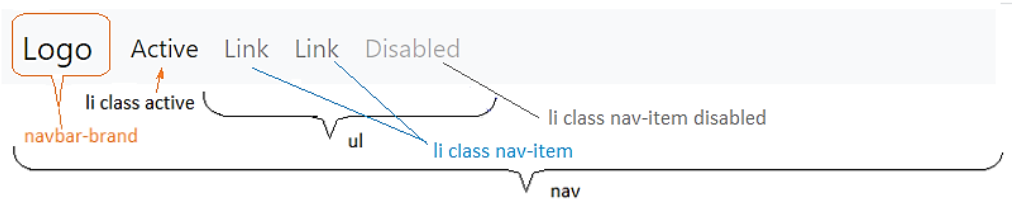
Meniuri

Cu Bootstrap, o bară de navigare se poate extinde sau colapsa, în funcție de dimensiunea ecranului.

O bară de navigare standard este creată cu `<nav class="navbar navbar-default">`.

Următorul exemplu arată cum să adăugați o bară de navigare în partea de sus a paginii:

```
<nav class="navbar navbar-expand-sm bg-light navbar-light">
  <a class="navbar-brand" href="#">Logo</a>
  <ul class="navbar-nav">
    <li class="nav-item active">
      <a class="nav-link" href="#">Active</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#">Link</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#">Link</a>
    </li>
    <li class="nav-item">
      <a class="nav-link disabled" href="#">Disabled</a>
    </li>
  </ul>
</nav>
```



Culorile meniului pot fi inversate dacă se înlocuiește în tag-ul nav clasele *bg-light navbar-light* cu *bg-dark navbar-dark*.

```
<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
```

Schimbarea paletei de culori se face alegând o culoare validă din clasele de culori Bootstrap pentru background:

```
<nav class="navbar navbar-expand-sm bg-primary navbar-dark">

<nav class="navbar navbar-expand-sm bg-warning navbar-light">
```

Contrastul dintre text și fundal se asigură alegând corect între clasele *navbar-light* și *navbar-dark*.

Figura 48 redă trei meniuri de același tip, cu background-uri de culori diferite.

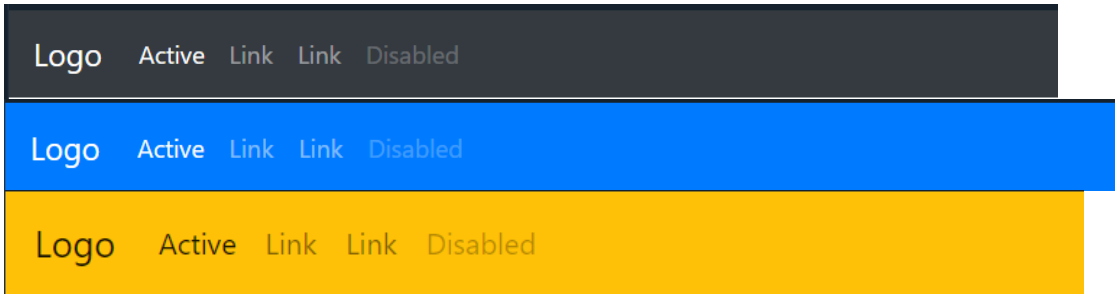


Figura 48. Diverse aspecte ale meniurilor: bg-dark, bg-primary și bg-warning

Clasa *navbar-expand-** este necesară pentru asigurarea responsivității prin rearanjarea meniului la dimensiunea specificată de * (sm, md, lg)

Figura 49 Meniu cu clasa navbar-expand-*

Pentru dispozitivele cu viewport-uri mici, cum ar fi smarphone-urile, se preferă colapsarea meniului într-o pictogramă, cunoscută sub numele de „hamburger”. Pentru a obține acest lucru trebuie utilizate și scripturile care completează biblioteca css:

```
<script
src="https://cdn.jsdelivr.net/npm/jquery@3.6.4/dist/jquery.s
lim.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/
popper.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/js/bo
otstrap.bundle.min.js"></script>
```

Apoi, întregul meniu se include într-un *div* cu clasa *collapse navbar-collapse*.

```
  <div class="collapse navbar-collapse" id="mainNavbar">
    <ul class="navbar-nav">
      <!-- Aici continutul meniului -->

    </ul>
  </div>
```

Mai este nevoie de butonul în care comută meniul, îl afișează sau îl ascunde la click:

```
<!-- Buton declansator/ascundere meniu →
     <button   class="navbar-toggler"   type="button"   data-
toggle= "collapse" data-target="#mainNavbar">
    <span class="navbar-toggler-icon"></span>
  </button>
```

Acest buton se plasează în afara div-ului care conține meniul propriu-zis. El conține două atribute de tip „custom”: *data-toggle* și *data-target*. În Bootstrap, *data-toggle* împreună cu valoarea *collapse* indică ce acțiune se execută la apăsarea butonului iar *data-target* indică asupra cărui element se execută acțiunea.

Exemplul complet poate fi văzut la adresa https://playcode.io/1593430.

Meniurile de tip drop-down se obțin ușor, adăugând clasa *dropdown* la elementul de tip <li> dorit din meniu, clasa *dropdown-toggle* la ancora elementului și atributul *data-toggle* cu valoarea *dropdown*:

```
<li class="nav-item dropdown">
  <a   class="nav-link   dropdown-toggle"   href="#"   data-
toggle="dropdown">
     Dropdown link
  </a>
… … …
</li>
```

Elementul <li> de tip *dropdown* conține un div cu clasa *dropdown-menu* care include un număr de ancore (link-uri) cu clasa *dropdown-item*:

```
<div class="dropdown-menu">
   <a class="dropdown-item" href="#">Item 1</a>
   <a class="dropdown-item" href="#">Item 2</a>
   <a class="dropdown-item" href="#">Item 3</a>
</div>
```

Exemplul anterior, modificat prin transformare ultimului link click-abil în unul de tip drop-down poate fi inspectat la adresa https://playcode.io/1593468.

Cele prezentate până aici oferă suficiente informații și instrumente pentru a ajuta un designer Web să genereze un layout cu un efort minim utilizând Bootstrap. Ele ar trebui să constituie și o provocare pentru aprofundarea studiului, care vă poate ajuta să creați formulare cu aspect plăcut, carusele de imagini, tooltip-uri etc. Documentația completă a Bootstrap poate fi consultată pe site-ul oficial

https://getbootstrap.com/docs/4.6/getting-started/introduction/, dar pentru început poate fi suficientă și cea de pe site-ul W3School https://www.w3schools.com/bootstrap4/

Concluzii

Am scris această carte cu gândul la studenții mei de la programul de studii Media digitală, care funcționează în cadrul Facultății de litere și științele comunicării din Universitatea Ștefan cel Mare, Suceava. În covârșitoare majoritate ei provin de la licee cu profil uman: filologie, pedagogic, științe sociale sau artistic. Abordarea conținutului de o manieră inginerească sau a informaticii teoretice este total neproductivă în ceea ce-i privește pe acești studenți. Din acest motiv am ales să transpun informația într-un limbaj cât mai accesibil, cu mai multă parte narativă și foarte multe exemple online. Codurile reproduse în carte pot fi reproduse și rulate prin simplă copiere și sunt complet funcționale.

Ce ar mai urma? Paginile create cu HTML și CSS sunt și rămân niște documente *statice*, singurele interacțiuni ale utilizatorului cu pagina rezumându-se la link-uri și formulare. Paginile moderne oferă mult mai mult și o experiență de navigare substanțial îmbogățită. Acesta este posibilă cu ajutorul unor programe executate în interiorul navigatorului, programe scrise în limbajul JavaScript. Prin urmare, următorul pas ar trebui să fie învățarea acestui limbaj și a câtorva framework-uri bazate pe JavaScript: React, VueJS etc.

Bibliografie

1. Bose, S. (2023, May 7). *Top 5 CSS Frameworks for Developers and Designers.* Preluat de pe Browserstack: https://www.browserstack.com/guide/top-css-frameworks
2. Featherly, K. (fără an). *ARPANET.* Preluat de pe Britannica: https://www.britannica.com/topic/ARPANET/A-packet-of-data
3. Fitzgerald, A. (2022, September 19). *14 Best HTML & CSS Code Editors for 2022.* Preluat de pe HubSpot: https://blog.hubspot.com/website/best-html-css-editor
4. Jacklin, B. (2023, mai 8). *What Is an OGG File and How to Play It.* Preluat de pe MOVAVI: https:/www.movavi.com/learning-portal/ogg-file.html
5. Pattakos, A. (2023, ianuarie 9). *9 Best CSS Frameworks in 2023.* Preluat de pe Athemes.com: https://athemes.com/collections/best-css-frameworks/