wantsome
The friendly IT Academy

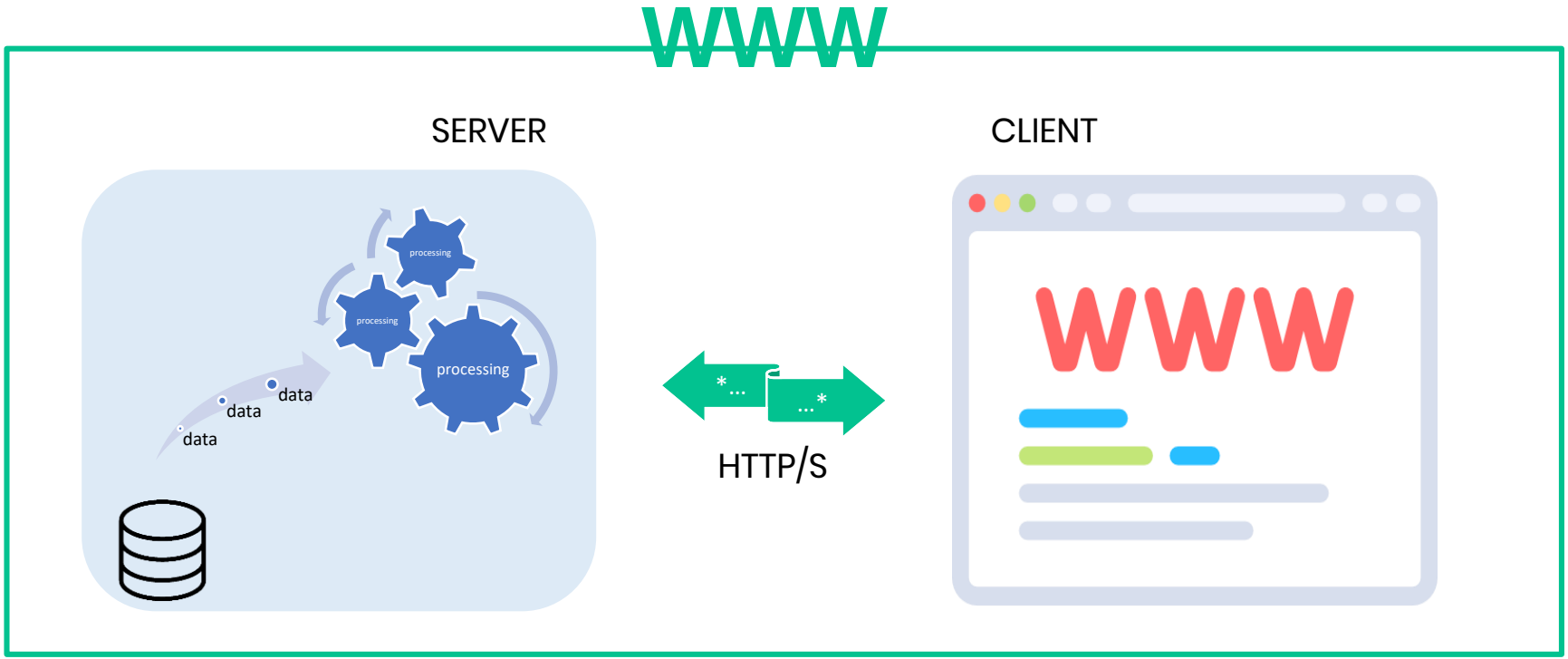
Saptamana 10

Partea 2

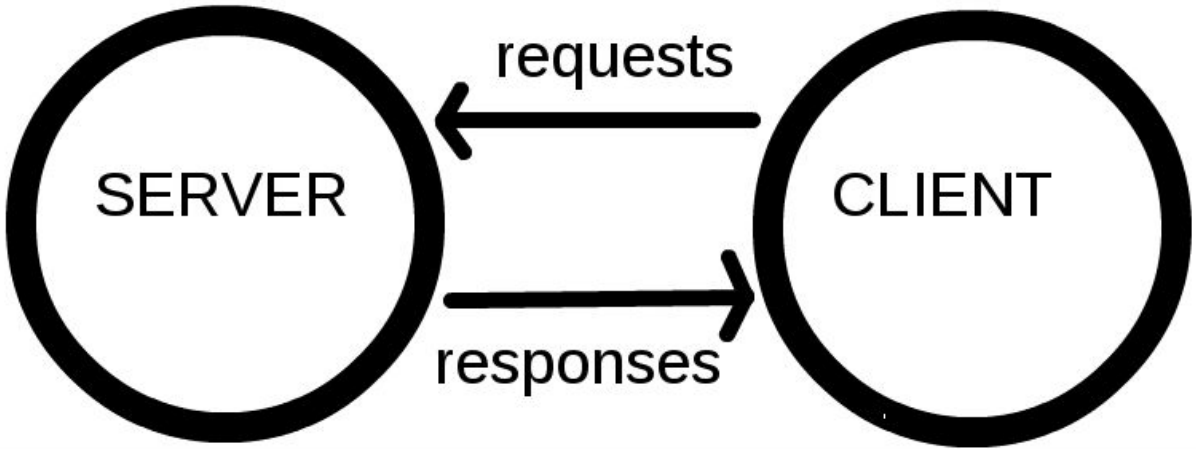
Programare Front-End

1. Web Apps Architecture Overview

Server - Retea - Client



Server - Retea - Client



Server - Retea - Client

Notiuni:

- **World Wide Web (WWW)**: Spatiu informatic in cadrul caruia diferite tipuri de documente si resurse sunt identificate de URLs (Uniform Resource Locators) accesibile prin intermediul Internetului

- **Server**: Calculator care stocheaza pagini web, siteuri sau aplicații, care operează mod continuu in cadrul unei retele, asteptand solicitări (cereri) din partea altor dispozitive din rețea (clienti); Când un client dorește să acceseze o pagină web, o copie a paginii web este descărcată de pe server pe mașina clientului pentru a putea fi afișată în browserul web al utilizatorului

  to serve = a servi

  BACK-END

- **Client:** Dispozitiv tipic conectat la internet al unui utilizator web (de exemplu, calculatorul tău conectat la Wi-Fi, sau telefonul tău conectat la rețeaua de telefonie mobilă) și softul de accesare a paginilor web disponibil pe acest dispozitiv (de obicei un browser web ca Firefox sau Chrome)

  FRONT-END

Server - Retea - Client

Notiuni:

- **Retea:** Canalul principal de comunicare intre ***Client - Server***

- **TCP/IP**: Transmission Control Protocol si Internet Protocol sunt protocoale de retea care definesc felul in care datele trebuie sa circule la nivel **WEB**; Primul controleaza felul in care se transmit pachetele ce alcatuiesc mesajele intre client si server, iar cel de-al doilea se ocupa cu asignarea de adrese pentru **clienti** si **servere** astfel incat sa se stie cine si cui trimite mesaje

- **DNS:** Server de nume de domeniu - “Pagini Aurii” pentru situri web. Când introduci o adresă web în browserul tău, acesta verifica numele de domeniu pentru a găsi adresa reală a sitului web înainte de a putea prelua situl. Browserul trebuie să afle pe ce server se află situl web, astfel încât să poată trimite mesaje de tip **HTTP** la locul potrivit

- **HTTP:** Hypertext Transfer Protocol definește un limbaj pentru **clienți** și **servere** pentru “a vorbi” între ei; Este construit peste **TCP/IP**

Server - Retea - Client

Notiuni:

- **Fisiere componente**: Un site web este alcătuit din mai multe fișiere diferite; Avem două tipuri principale:
    - **Fișiere cod:** Siteurile web sunt construite în principal din **HTML**, **CSS** și **JavaScript**, dar mai tarziu vom intalni si alte tehnologii
    - **Resurse**: Fisiere media precum imagini, muzică, video, documente Word, fișiere PDF, etc...

Ce se intampla mai exact ?

1. Browserul comunica cu serverul DNS pentru a gasi adresa reala a serverului unde se afla gazduit *site*-ul web

1. Browserul trimite un mesaj de tip **HTTP,** cerandu-i acestuia o copie a documentului principal care reprezinta *site*-ul (**index.html**). Acest mesaj, impreuna cu toate celelalte date trimise intre *server* si *client* sunt transmise prin intermediul internetului, utilizand protocolul TCP/IP

1. Daca serverul accepta cererea de la client, acesta raspunde cu un mesaj care are ca si cod de stare "**200 OK**", iar mai apoi incepe sa trimita fisiere catre browser sub forma mai multor pachete de date de dimensiuni mai mici

1. Browser-ul stie sa asambleze aceste pachete mici de date si sa alcatuiasca site-ul web care este afisat in fereastra sa

Referinta: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/How_the_Web_works

2. HTTP

HTTP

- **HyperText Transfer Protocol**

- protocol de retea, **client - server**

- de fiecare care data cand “navigam” pe *web, browser*-ele vor trimite mesaje de tip ***request*** *HTTP* pentru pagini *HTML*, imagini, *scripts*, foi de stiluri, samd… (**resurse**)

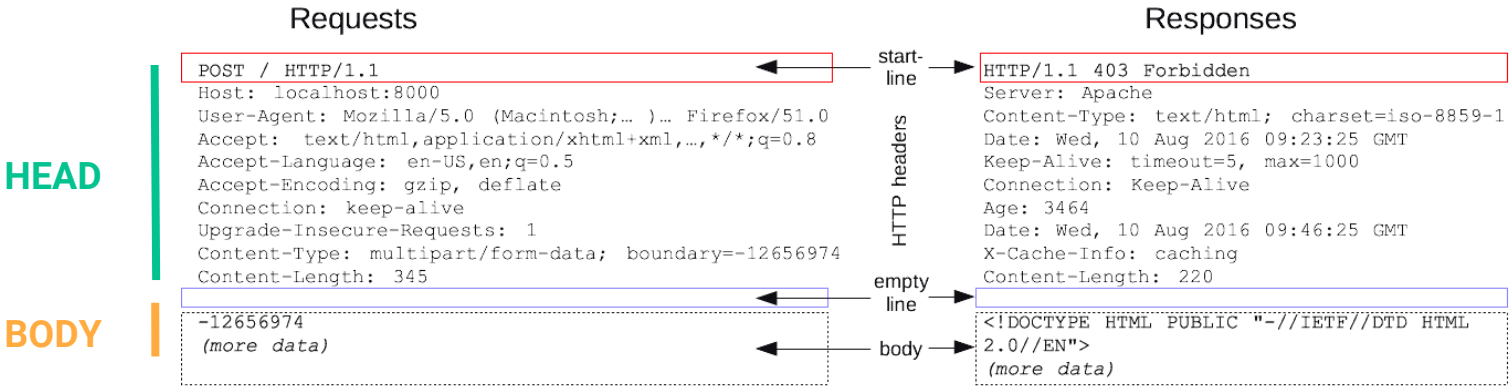
- *server*-ele web trateaza aceste *request*-uri returnand mesaje de tip ***response*** care contin resursele “cerute” de catre *browser*-e

2. HTTP Messages

HTTP Messages

Mesajele HTTP de tip **request** si **response** au in comun aceeasi structura si sunt compuse din:

1. O linie de inceput care descrie tipul mesajului (**metoda**), alaturi de un **cod de stare**
2. Un set optional de **header**-e care descriu mesajul si continutul **body**-ului
3. O linie fara continut text (**blank line**) care separa **head**-ul de **body**
4. Un **body** optional care contine date asociate cu mesajul (precum datele completate intr-un formular *HTML* de catre utilizator) sau, spre exemplu, o resursa de tip fisier asociata cu mesajul - existenta si tipul de continut al **body**-ului sunt specificate in **head**



HTTP Methods

- pentru a defini felul in care resursele transmise prin intermediul mesajelor *HTTP* trebuie tratate, sau care este actiunea care se doreste a fi executata asupra resursei in cauza, *HTTP* are la baza, prin definitie, un set de **metode** care specifica acest lucru

- de cele mai multe ori, aceste metode sunt intalnite si sub denumirea de **HTTP verbs**

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

! Obs.: Parcurgeti conceptele de **metode safe**, **idempotent**-e si **cacheable** din resursa specificata mai sus

HTTP Status Codes

- codurile de stare indica statusul unui request creat de catre browser catre server

- sunt alcatuite din 3 cifre, iar prima cifra a fiecarui cod este una din setul [1,2,3,4,5] - deci exista 5 clase de statusuri
  - **1**xx - statusuri informationale
  - **2**xx - operatiuni produse cu succes
  - **3**xx - redirectari
  - **4**xx - erori la nivel de client
  - **5**xx - erori la nivel de server

https://httpstatuses.com/

HTTP Requests and HTTP Responses

https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages#HTTP_Requests

3. AJAX

... nu detergentul.

si nici echipa de fotbal


AJAX

AJAX Intro

- **A**synchronous **J**avaScript **a**nd **X**ML

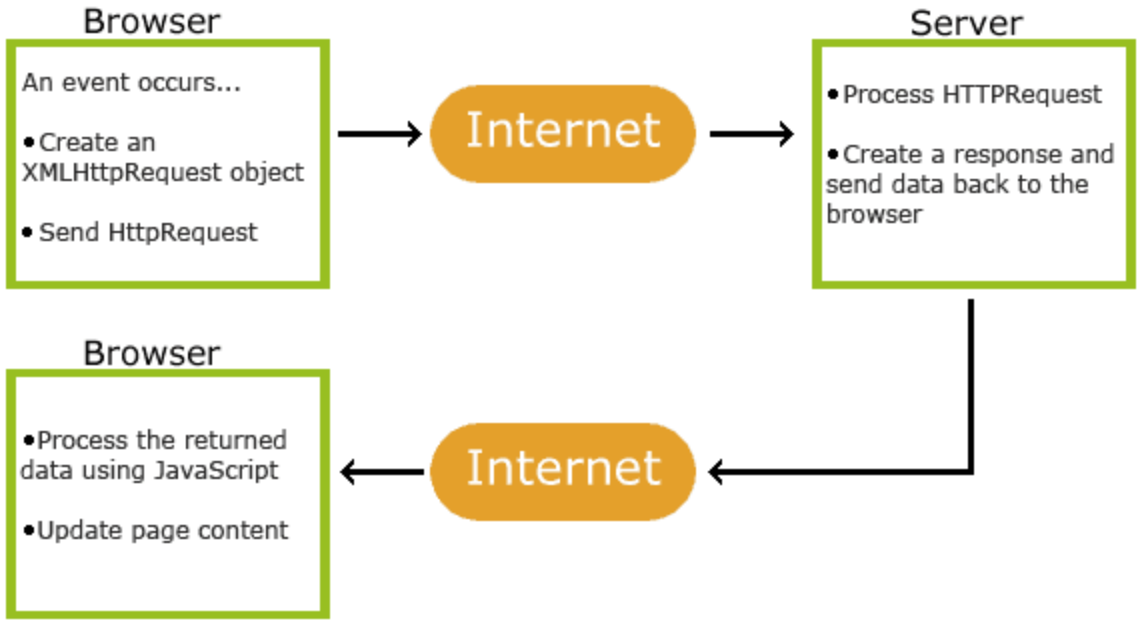
- nu este un limbaj de programare ci o metodologie

- permite:
  - actualizarea datelor din cadrul unei pagini web fara a fi nevoie reincarcarea acesteia
  - crearea de *requests* catre server dupa incarcarea paginii
  - primirea de *responses* de la server dupa incarcarea paginii
  - trimiterea de date catre server in *background*, in timp ce utilizatorul interactioneaza cu pagina

AJAX Intro

- se bazeaza pe:
    - **XMLHttpRequest** - obiect pus la dispozitie de catre browser pentru comunicarea cu servere web
    - **JavaScript** si **DOM Manipulation** - pentru utilizarea datelor introduse de catre utilizator sau popularea interfetei cu date primite de la un server

AJAX Intro



XMLHttpRequest - Intro

- toate browser-ele moderne suporta obiectul **XMLHttpRequest**

- asigura comunicarea cu serverele web si popularea cu date a interfelor aplicatiilor web fara a fi necesara reincarcarea paginilor

```
var xhttp = new XMLHttpRequest();
```

4. XMLHttpRequest

XMLHttpRequest - Exemplu

```
function getAjaxInfoTextDocument() {
  var xhttp = new XMLHttpRequest(); // initializarea unei noi instante a obiectului
  xhttp.onreadystatechange = function() { // declararea functiei care trateaza raspunsul
    if (this.readyState == 4 && this.status == 200) { // verificarea starii - cazul OK
          document.getElementById("demo").innerHTML = this.responseText; // populatea cu date
    }
  };
  xhttp.open("GET", "ajax_info.txt", true); // specificarea metodei si a locatiei server-ului
  xhttp.send(); // trimiterea request-ului
}
```

XMLHttpRequest - Metode si proprietati

https://www.w3schools.com/xml/ajax_xmlhttprequest_create.asp

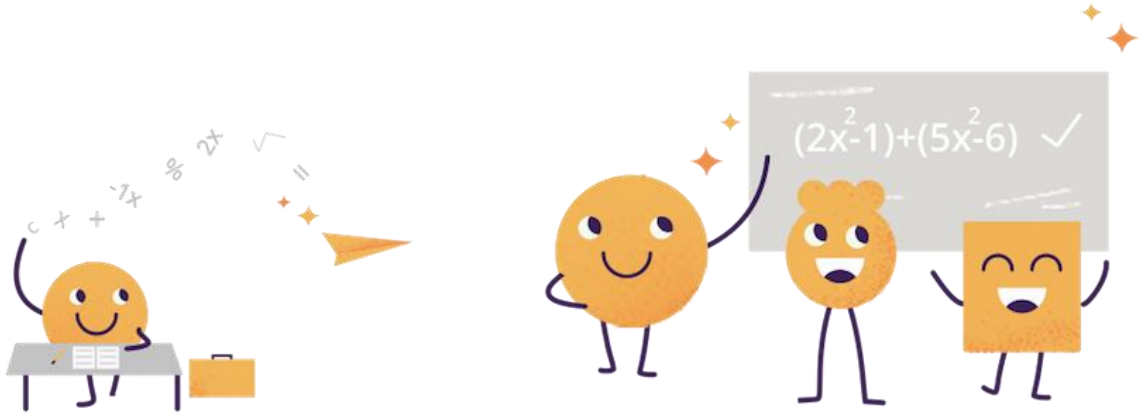
XMLHttpRequest - JSON Data Format

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON

PRACTICE: **XMLHttpRequest and JSON Data Format**

http://bit.do/json-example

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON#Active_learning_Working_through_a_JSON_example

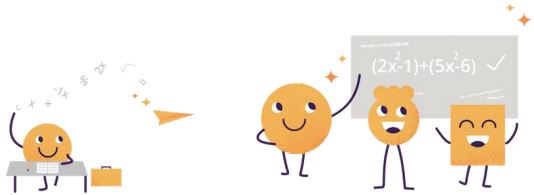


PRACTICE: **XMLHttpRequest and JSON Data Format**

Cerinta:

*La adresa **http://bit.ly/json-colors-practice** veti gasi un JSON care contine o lista de culori in formatul **"nume-culoare": "HEX-code"**. Creati o pagina care sa afiseze toata lista de culori in felul urmator: fiecare linie a unei liste este asociata unei culori; o linie corespunzatoare unei culori va contine un patrat de dimensiuni 10x10 care are ca si background culoarea respectiva; langa patrat, este afisat numele culorii; folositi CSS pentru alinieri si afisati textul centrat pe verticala in raport cu patratul culorii; lista nu trebuie sa contina bullet points;*

1. *Initial, veti stoca toata lista JSON intr-o variabila prin intermediul careia veti parcurge datele*
2. *Lista JSON va fi descarcata de la URL-ul corespunzator folosind **XMLHttpRequest** si o veti stoca ulterior in variabila anterior creata la pasul 1*



PRACTICE: **XMLHttpRequest and JSON Data Format**

http://bit.do/exXHR

http://bit.do/ex2XHR