TRANSMISIA DATELOR - APLICATII

Descrierea mediului de lucru

CUPRINS

1 Descrierea mediului de lucru 3
1.1 Obiective . . . 3
1.2 Introducere . . . 3
1.2.1 Motivație . . . 3
1.2.2 HTML . . . 3
1.2.3 CSS . . . 4
1.2.4 JavaScript . . . 4
1.2.5 Unelte de dezvoltare . . . 6
1.3 Desfășurarea lucrării . . . 6
1.3.1 Prima aplicație JavaScript . . . 6
1.3.1.1 Instrucțiuni uzuale JavaScript . . . 7
1.3.2 Implementarea și testarea unei funcții în JavaScript . . . 9
1.3.3 Proiectarea unei interfețe grafice în HTML . . . 10
1.3.4 jQuery . . . 12
1.4 Aplicații și probleme . . . 14
1.4.1 Problema 1 *(3 puncte)* . . . 14
1.4.2 Problema 2 *(1.5 puncte)* . . . 14
1.4.3 Problema 3 *(1.5 puncte)* . . . 14
1.4.4 Problema 4 *(2 puncte)* . . . 14
1.4.5 Problema 5 *(2 puncte)* . . . 15
1.5 Ce vom învăța în laboratorul următor? . . . 15

Bibliografie 16

1 DESCRIEREA MEDIULUI DE LUCRU

1.1 OBIECTIVE

- Introducere în tehnologiile HTML, CSS și JavaScript
- Proiectarea și implementarea interfețelor grafice
- Gestionarea evenimentelor în aplicații software

1.2 INTRODUCERE

1.2.1 MOTIVAȚIE

În prezent pachetul de tehnologii ("technology stack") **HTML, CSS și JavaScript** este extrem de popular în industria IT pentru că se adresează unui spectru larg de aplicații software. Cu ajutorul acestor tehnologii se dezvoltă în special aplicații web. Totuși, în ultima perioadă, datorită apariției unor platformele cum sunt Node.js [1] și Electron [2]), există un sprijin tot mai puternic pentru aplicații backend, desktop și mobile.

Un citat celebru din anul 2007 spune că *"orice aplicație care poate fi scrisă în JavaScript va fi în cele din urmă scrisă în JavaScript"* - Jeff Atwood, Cofondator al StackOverflow. Acest fapt este confirmat de ultimele sondaje care se referă la cele mai frecvent utilizate limbaje de programare. JavaScript este limbajul de programare cel mai popular atât pe pe Github cât și pe StackOverflow, după cum reiese din statisticile pe anul 2018 prezentate în Figurile 1.1 și 1.2.

De asemenea, multe companii importante din industria software (Facebook/React, Google/Angular, Microsoft, Mozilla, Yahoo, Alibaba) susțin și contribuie la dezvoltarea uneltelor și librariilor necesare pentru dezvoltarea aplicațiilor pe aceste trei limbaje.

1.2.2 HTML

Acronimul HTML vine de la "Hyper Text Markup Language" si reprezintă un limbaj care ne permite sa definim interfete grafice pentru aplicații software care sunt executate în browser.

O pagină HTML are o structură sub formă de arbore formată din elemente grafice, care are nod rădăcină elementul <html>. Acesta are doi copii <head> și <body>. În <head> se descrie pagina web (meta data, titlu, tipul de caractere folosite). De asemenea, se includ eventualele fișiere externe cu stiluri CSS și scripturile JavaScript. În <body> se descrie interfața grafică care într-o aplicație web include: antet, meniu și conținut.

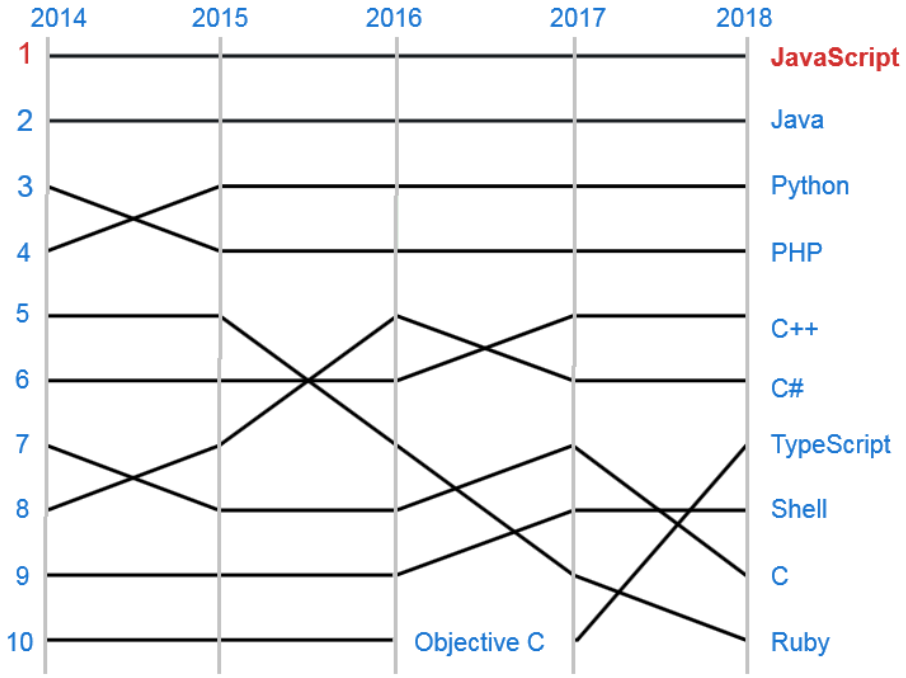

Figura 1.1: Limbajele de programare cele mai populare pe Github în 2018 (Sursa: [3])

Printre elementele HTML cele mai importante se numără: , <input> și <button>.

1.2.3 CSS

CSS (Cascading Style Sheets) este limbajul prin care se descrie modul în care elementele HTML sunt reprezentate grafic. Elementele grafice au un stil standard de afișare care este specific browserului în care pagina este afișată. Totuși, acest stil poate fi suprascris și customizat folosind sintaxa CSS.

Câteva proprietăți CSS frecvent utilizate sunt: font-size, border, background-color, color, width și height.

Cu ajutorul stilurilor definite în CSS se poate controla modul de reprezentare grafică al aplicațiilor pe diferite medii de redare (monitoare, laptopuri, smartphone-uri, hârtie).

1.2.4 JavaScript

JavaScript este un limbaj de programare creat de către Brendan Eich în 1995 și a fost folosit pentru prima dată în browserul Netscape.

JavaScript este limbajul care controlează comportamentul interfețelor grafice web și

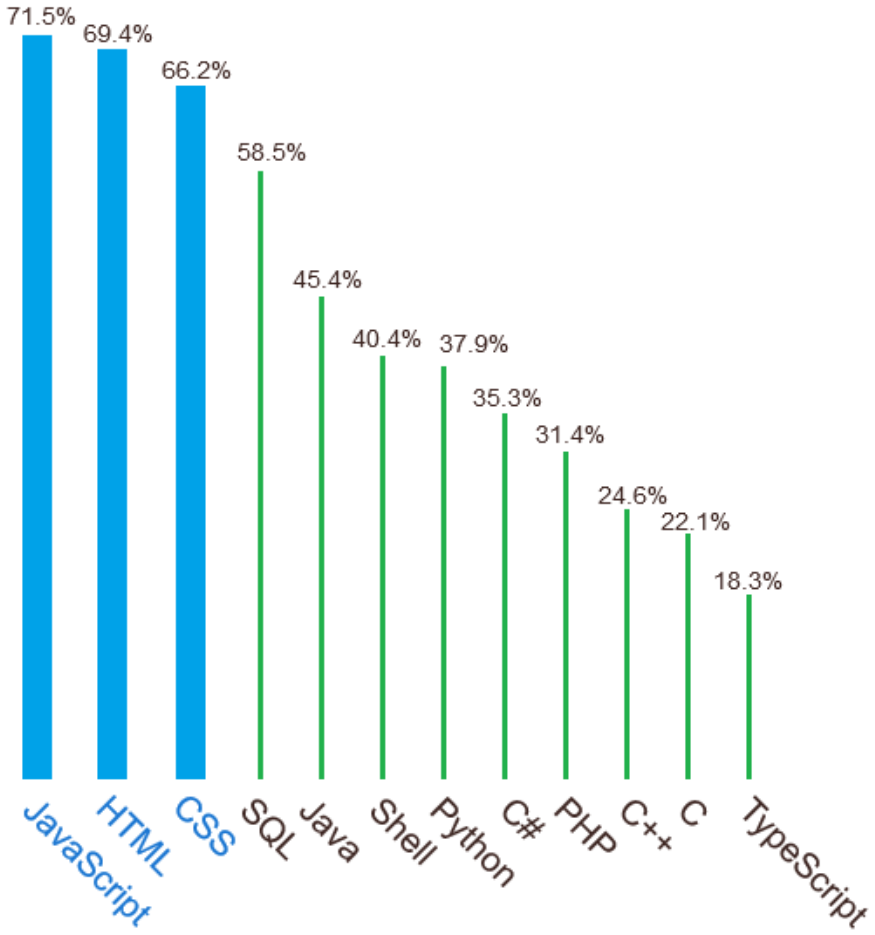

Figura 1.2: Limbajele de programare cele mai populare pe StackOverflow în 2018 (Sursa: [4])

nu numai. Chiar dacă anterior JavaScript-ul era folosit în special pentru aplicații web, în ultimii ani a devenit o alegere tot mai populară pentru aplicații desktop și mobile.

Există mai multe librarii care permit crearea unor aplicații client complexe: jQuery, React, VueJS, Angular. Acestea extind capabilitățile JavaScript prin diverse funcționalități uzuale cum ar fi: sintaxă simplificată pentru referențierea elementelor HTML, legătură bidirecțională între model (variabile) și interfața grafică, arhitectură bazată pe componente.

Pentru orice tip de aplicații (frontend, backend, mobile) în general se folosește platforma NodeJS care facilitează structurarea și automatizarea multor pași necesari în dezvoltarea

software (importul de librării, execuția unor servere web care servesc resurse, configurarea și execuția buildului pe client).

1.2.5 UNELTE DE DEZVOLTARE

Deși browserul și Notepad-ul sunt suficiente pentru a scrie și testa aplicații simple, o dată ce programele devin mai complexe se recomandă utilizarea următoarelor editoare de cod: Visual Studio Code, Sublime sau Atom.

1.3 DESFĂȘURAREA LUCRĂRII

1.3.1 PRIMA APLICAȚIE JAVASCRIPT

Prima aplicație conține 3 fișiere: **index.html**, **style.css** și **app.js**. Fișierul **index.html** definește structura interfeței grafice și care pentru început va afișa un simplu câmp text. În secțiunea <head> se include fișierul CSS iar la sfârșitul secțiunii <body> fișierul JavaScript.

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4      </head>
5      <body>
6          <div id="message"></div>
7
8          <script src="app.js"></script>
9      </body>
10 </html>
```

Este important ca instrucțiunile JavaScript să fie executate doar în momentul în care toate elementele grafice HTML din <body> au fost deja încărcate în browser. Acest lucru se poate realiza prin amplasarea scriptului la sfîrșitul elementului <body>, având în vedere că resursele (JS, CSS, HTML) se încarcă în browser secvențial în ordina apariției lor în fișierul HTML. O altă soluție este executarea codului doar la apariția evenimentului generat când pagina s-a încărcat (ex: <body onload="run();">).

În fișierul JavaScript **app.js** se afișează la consola din browser un mesaj:

```
1  console.log("Welcome to data transmission");
```

> ☞ Pentru executa aplicația și a vedea mesajul din JavaScript se deschide **index.html** în orice browser, se apasă tasta F12 și se selectează tabul **Console**.

În continuare se va modifica din JavaScript valoarea șirului de caractere afișat în elementul HTML astfel:

```
1 document.getElementById("message").innerHTML = "Message from
    JavaScript";
```

Cu ajutorul obiectului nativ JavaScript denumit **document** [5] se poate obține o referință spre orice element HTML be baza unui selector, fie acesta tipul elementului, id-ul sau clasa acestuia. În momentul în care obținem această referința putem controla caracteristicile elementului HTML din JavaScript. De asemenea, putem gestiona anumite evenimente care sunt generate atunci când un utilizator folosește interfața grafică.

> ☞ Identificatorul folosit în JavaScript ca argument al metodei getElementById ("**message**") este identic cu identificatorul unic specificat în elementul HTML " ".

În momentul în care obținem referința spre elementul HTML din JavaScript, avem posibilitatea să modificăm caracteristicile acestuia. De asemenea, putem gestiona anumite evenimente care sunt generate atunci când utilizatorul interacționează cu obiectul grafic.

1.3.1.1 INSTRUCȚIUNI UZUALE JAVASCRIPT

În JavaScript variabilele se declară utilizând cuvântul cheie **var** pus în fața numelui variabilelor și pot lua diferite valori. În scriptul următor se vor declara patru tipuri de variabile (Number, String, Boolean și Object).

```
1 var sum = 10;
2 var name = "Alexandru";
3 var isActive = true;
4 var user = {id: 1, name:"Andrei", age: 21, };
```

Obiectul user este în format JSON (JavaScript Object Notation). În sintaxa JSON obiectele încep cu { și se termină cu }. Între acolade se specifică perechi de tipul **cheie: valoare** care definesc caracteristicile specifice ale obiectului. Obiectele JSON pot include și liste de elemente care se declară între paranteze drepte. Un exemplu de obiect JSON complex este prezentat în scriptul următor:

```
1  var user = {
2     "id": 1,
3     "name": "Alexandru Cristea",
4     "username": "acristea",
5     "email": "acristea@test.com",
6     "address": {
7       "street": "Padin",
8       "number": "Ap. 10",
9       "city": "Cluj-Napoca",
10      "zipcode": "123456",
```

```
      "geo": {
        "lat": "46.783364",
        "lng": "23.546472"
      }
    },
    "phone": "004-07xx-123456",
    "company": {
      "name": "XYZ",
      "domain": "Air Traffic Management",
      "cities": ["Cluj-Napoca", "Vienna", "Paris"]
    }
  }
```

Afișarea diferitelor caracteristici ale obiectului user este exemplificată în următorul script:

```

console.log(user.name);
console.log(user.address.geo.lat);
console.log(user.company.name);
console.dir(user.company.cities);
console.log(user.company.cities[0]);
...
```

În continuare vom defini și apela o funcție:

```
function print(message){
    console.log(message);
}
print("hello");
```

Operatorul ternar este o simplificare a instrucțiunii **if** și este exemplificat în continuare:

```
var password="123456";
console.log(password=="123456"?"ALLOW":"DENY");
```

Logica echivalentă descrisă cu instrucțiunea **if** este următoarea:

```
var password="123456";
if(password == "123456"){
    console.log("permission accepted");
} else {
    console.log("permission accepted");
}
```

1.3.2 IMPLEMENTAREA ȘI TESTAREA UNEI FUNCȚII ÎN JAVASCRIPT

Aplicația conține 3 fișiere: **index.html app.js** și **appTest.js**. Interfața grafică ne permite să introducem o valoare într-un câmp input:

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4      </head>
5      <body>
6          <input type="text" id="n"  value="5"/>
7
8          <script src="app.js"></script>
9          <script src="appTest.js"></script>
10     </body>
11 </html>
```

Fișierul **app.js** conține:

```
1  document.getElementById("n").addEventListener('input',
       inputSum);
2
3  function inputSum(){
4      var inputNumber = parseInt(document.getElementById("n").
           value);
5      sum(inputNumber);
6  }
7
8  function sum(n){
9      if (typeof n === 'undefined') return "n is undefined";
10     var sum = 0;
11     for(var i=1;i<=n;i++){
12         sum+=i;
13     }
14     return sum;
15 }
```

- Linia 1: Se specifică un event handler care va fi apelat atunci când utilizatorul modifică valoarea din input.
- Linia 4: În JavaScript variabilele se definesc folosind cuvântul cheie **var**. Se citește valoarea din câmpul input și se convertește din șir de caracter în număr întreg.
- Linia 5: Se apelează funcția care returnează suma elementelor de la 1 până la n.

Fișierul **appTest.js** conține o suită de 4 teste care validează comportamentul dorit al funcției **sum**:

```
function test(){
    console.log(sum(0)==0?"Passed":"Failed");
    console.log(sum(2)==3?"Passed":"Failed");
    console.log(sum(4)==10?"Passed":"Failed");
    console.log(sum()=="n is undefined"?"Passed":"Failed");
}
test();
```

Sintaxa JavaScript ne permite să scriem codul de mai sus și ca funcție anonimă executată imediat astfel:

```
(function(){
    console.log(sum(0)==0?"Passed":"Failed");
    console.log(sum(2)==3?"Passed":"Failed");
    console.log(sum(4)==10?"Passed":"Failed");
    console.log(sum()=="invalid argument"?"Passed":"Failed");
})();
```

1.3.3 Proiectarea unei interfețe grafice în HTML

Fișierul **index.html** definește structura interfeței grafice care conține un label text, și un buton.

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" href="style.css">
    </head>
    <body>
        <div id="counter"></div>
        <button id="inc">+</button>

        <script src="app.js"></script>
    </body>
</html>
```

În continuare se va modifica stilul CSS din fișierul **style.css** astfel:

```
body {
  padding: 20px;
  text-align: center;
  background-color:#f5f5f5;
}
```

```
6
7  #widget {
8    width: 60px;
9    background-color: #555;
10   margin: 0 auto;
11   padding: 10px;
12 }
13
14 #counter {
15   height: 50px;
16   width: 50px;
17   margin: 0 auto;
18   background-color: #fff;
19   color: #555;
20   text-align: center;
21   line-height: 50px;
22   font-size: 30px;
23 }
24
25 #inc {
26   margin-top: 10px;
27   height: 50px;
28   width: 50px;
29   background-color: #5b9aff;
30   color: #fff;
31   text-align: center;
32   line-height: 50px;
33   font-size: 20px;
34   border: 0px;
35 }
36
37 #add:hover {
38   cursor: pointer;
39   background-color: #8cb8ff;
40   color:#fff;
41 }
```

Fișierul JavaScript **app.js** descrie comportamentul interfeței grafice și anume logica necesară ca valoarea contorului să fie incrementată de fiecare dată când butonul este apăsat de către utilizator.

```
1 var counter = 0;
2
3 function printValue(divId, value){
```

```
4   document.getElementById(divId).innerHTML = value;
5 }
6 printValue("counter", 0);
7
8 document.getElementById("inc").addEventListener("click",
     increment);
9
10 function increment(){
11   counter++;
12   printValue("counter", counter);
13 }
```

- Linia 1: Starea aplicației.
- Linia 3: Funcția care ne permite să afișăm o valoare text într-un div cu un identificator specificat.
- Linia 6: Afișarea inițială a contorului pe interfața grafică.
- Linia 8: Specificarea unui event handler care va fi apelat atunci când utilizatorul apasă butonul de incrementare. Acest event handler conține 2 parametri: tipul evenimentului și funcția care va fi apelată la producerea evenimentului specificat.

1.3.4 jQuery

jQuery este o librarie scrisă în JavaScript care simplifică și oferă o sintaxa mai expresivă de programare și extinde totodată funcționalitățile limbajului JavaScript. jQuery se poate include în secțiunea head a paginii HTML astfel:

```
1 <head>
2     <script src="https://ajax.googleapis.com/ajax/libs/jquery
         /3.3.1/jquery.min.js"></script>
3 </head>
```

Pentru obținerea referinței spre un element HTML se folosește sintaxa simplificată **$("#id")**. Simbolul "$" specifică începutul unui obiect/funcții jQuery. Codul JavaScript din secțiunea 1.3.3 poate fi rescris cu jQuery astfel:

```
1 var counter = 0;
2
3 function printValue(divId, value){
4     $("#"+divId).html(value)
5 }
6 printValue("counter", 0);
7
```

```
8  $("#inc").on('click', increment);
9
10 function increment(){
11   counter++;
12   printValue("counter", counter);
13 }
```

Linia 4: Pentru a scrie într-un element de tip **div** se folosește funcția **html()** din libraria **jQuery**.

Dacă se dorește utilizarea unui câmp de tip **input** text în aplicație, citirea (linia 1) și setarea (linia 2) valorii elementului **input** cu jQuery se face astfel:

```
1 var textValue = $('#inputTextId').val();
2 $('#inputTextId').val('123');
```

1.4 APLICAȚII ȘI PROBLEME

1.4.1 PROBLEMA 1 *(3 puncte)*

Să se implementeze/testeze toate aplicațiile descrise în Secțiunea 1.3:

- Modificarea valorii unui câmp text din JavaScript folosind funcția **getElementById** accesibilă din interfața obiectului **document** *(0.25 p)*.
- Afișare mesaj la consolă din JavaScript *(0.25 p)*.
- Accesarea unei valori dintr-un obiect JSON *(0.25 p)*.
- Definirea și apelul unei funcții JavaScript *(0.25 p)*.
- Utilizarea instrucțiunii **if** ca operator ternar *(0.25 p)*.
- Proiectarea unei interfețe grafice în HTML *(0.5 p)*.
- Includerea librariei jQuery în aplicașia JavaScript *(0.25 p)*.

1.4.2 PROBLEMA 2 *(1.5 puncte)*

Să se extindă aplicația prezentată în Secțiunea 1.3.1 prin adăugarea unui buton de decrementare și să se restricționeze valoarea contorului pe intervalul [0-10].

1.4.3 PROBLEMA 3 *(1.5 puncte)*

Să se modifice logica funcției **sum** din Secțiunea 1.3.2 astfel încât să permită exclusiv introducerea unui număr ca argument de intrare. De exemplu, dacă se introduce un șir de caractere se așteaptă returnarea valorii "not a number".

Să se scrie două teste care verifică introducerea unui șir de caractere și o variabilă booleană ca argument al funcției sum.

1.4.4 PROBLEMA 4 *(2 puncte)*

Să se scrie o funcție JavaScript cu antetul **getFibonacci(n)** care generează elemente din șirul lui Fibonacci până la o limită impusă **n** și să se scrie teste care validează comportamentul următor:

- Funcția returnează șirul de numere [1, 1] atunci când argumentul de intrare este 2.
- Funcția returnează șirul de numere [1, 1, 2, 3, 5] atunci când argumentul de intrare este 5.

- Funcția apelată fără argument sau cu orice tip de dată în afară de număr returneaza "not allowed".
- Funcția apelată cu n<1 sau n>10 returneaza "not allowed".

1.4.5 PROBLEMA 5 *(2 puncte)*

Pe baza modelului prezentat în Secțiunea 1.3.2 să se implementeze cu ajutorul **jQuery** interfața grafică pentru un calculator simplu care permite operații de adunare, scădere, înmulțire, împărțire și rest [+, -, /, *,
Câmpuri necesare:

- Două câmpuri **input** text (cu id-urile **firstNumber** și **secondNumber**) pentru preluarea datelor de la utilizator. În jQuery se folosește funcția **val** a elementului **input** HTML (ex: var firstNumberText = $('#firstNumber').val();) pentru a citi valoarea câmpului text. Pentru conversia din șir de caractere în număr întreg se poate folosi funcția **parseInt** (ex: var firstNumber = parseInt(firstNumberText)).
- Un element **button** pentru egal (cu event handler la click).
- Afișarea rezultatului într-un **div**.

1.5 CE VOM ÎNVĂȚA ÎN LABORATORUL URMĂTOR?

Despre:

- Medii de transmisie cu fir.
- Transferarea datelor între componente software distribuite utilizând JavaScript atât pe frontend (UI) cât și pe backend (API).
- Utilizarea protocoalelor **http** și **websocket** pentru dezvoltarea de aplicaţii client-server.

Vom folosi librăriile şi tehnologiile **NodeJS**, **VueJS** și **Express.js**.

Bibliografie

[1] https://nodejs.org/en/

[2] https://electronjs.org

[3] https://octoverse.github.com/projects#languages

[4] https://insights.stackoverflow.com/survey/2018/#technology

[5] https://www.w3schools.com/js/js_htmldom.asp