



HTML

(HyperText Markup language)

• fit me

- html tag : < button> , <p>, etc.
 - tag name: button, p, etc.

HTML2 attribute (modify how an elem. behaves)

```
<a href = " .. " > .. </a>
```

↓
anchor elem. (link to another website)

$$\text{freq} = \frac{\text{attn.}}{\text{attn. value}}$$

- opening tag: <-->
 - closing tag: </-->
 - content tag: <--> ↓ </-->

Attributes list

- `target = " "`
 - `target = "blank"` → open in new tab

extra spaces and lines

are ignored in HTML

CSS Basics

(Caseading Style Sheets)

= change the appearance of an HTML page

<Style> - - - </style>

exemplu: button \rightarrow - - - - - CSS selector (which elem.)

background-color: red; -- --- CSS property (what we're changing)

} -- CSS rule

CSS properties

- color → text color
 - border : none → remove the border
 - height (in px)
 - width (in px) → lengthwise
 - border-radius (in px) → rounding each corner
 - cursor: pointer → in local coordinate space
 - border-color
 - border-style (solid, dotted, -)
 - font-size (in px)

RGB (red, green, blue)
→ C C
 $\in [0, 255]$

Mong 'm = go around

- vertical-align: top;
 - border-width (in px)
 - margin-right (in px)
 - font-weight: bold

<button class = "button - join"> ... </button>

In style:

- button - join ↴
- :
- }

! pt a face pt 2 elem la file
(ex. button) ne pt face close

Hover, Transition, Shadows

.main-class : hover ↴
background-color : green;
}

.main-class : active ↴
--
}

! Add prevent transitions in class hover → fade out layer

Shadow box-shadow: 0 0 0 color
 ↑
 ↓ (black, etc)
 horizontal ↴ blur
 position
 (in px)

CSS Properties

- opacity (0-1)
- transition: opacity 1s;

Text Styles

- font-family : Arial;
- font-size : 30px;
- font-weight : bold;
- font-style : italic; (see underline)
- text-align: center; (left, right, justify)
- width; (each comma, and font size text per paragraph)
- line-height : 3px; (dist. dividers lines)
- background-color (padding)

pattern include content . specify to text

ex: 3.4 ml: · da
↓
adres *

pattern folia : hover

= modify style auto text
dimin - um paragraf

In <p> -- </p> :

<u> -- </u> → underline

 -- → bold

HTML Structure

```
<!DOCTYPE html>  
<html>  
<head> ... </head>  
<body> ... </body>  
</html>
```

In header

- title
- style (or CSS file)

CSS file: <link rel="stylesheet" href="styles.css" /> → void func.

Tags in body

- a, p, br, hr, h1, h2, h3
- img, src, height, width
- div, span
- br
- i → em
- table, tr, td, th, colspan, rowspan
- iframe → incorporate another HTML file

Deprecated (HTML 4)

- bgcolor, color
- font, size
- center

Images and text boxes

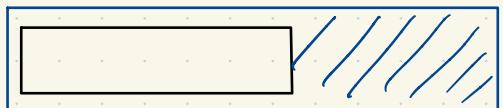
```
  
<input type="text" placeholder="search">  
    ↳ textbox  
    "checkbox"
```

CSS

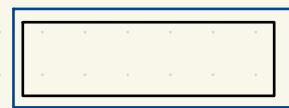
```
.logo {  
    width: 20px;  
    height: 60px;  
}  
  
↓  
relative ↳  
    - container  
        ↳  
        - content  
            ↳  
            - position: absolute;  
            ↳  
            - object-fit: cover;  
            ↳  
            - object-position: left;
```

CSS Display Property

- block elem = takes up the entire line



- inline block elem. = takes up as much as they need



- inline elem = text elem (ex: <p> -- -- -- </p>

In CSS : display : inline-block;

In CSS:

- margin,
- margin 2 b
- -
- 3

Div Element (division)

↳ just a box

<div> .. </div>

- use container div elem.

Nested Layout Technique

- vertical / horizontal layout

create 2 divs for horizontal layout (inline block elem), set w & margin

CSS Grid

<div style = " display: grid;

→ m. div val det. m. de col/row

grid-template-columns: 100px 100px;

">

<div> div 1 </div>

<div> div 2 </div>

</div>

data adding 2 de 1fr

⇒ 2 col. equal

- separate grid in grid
- max min col
- col. → new row

grid-template-columns: 100px 1fr;

→ take the remaining space

- column-gap: 2px;

- row-gap: 3px;

Flex Box

= another way to create layouts

display: flex;

flex-direction: row; → align-items: horizontal

la nivelul div-urilor: style = "flex: 1;"

style = "flex: 2;"

justify-content: center; → alinișă tot conținutul din div la centru

space-between; → equal amount of space

align-items: stretch; → cum se alinișă vertical

start 

end 

center

CSS Position

- static → default
- fixed → sticked ⇒ prob la div ⇒ - top: 0px; (dist de susul panou la div)

- left: 0px;

- right: - -

- height

- absolute → acum ea cel fixed

Diferența: fixed - placed in browser window

absolute - placed on the page

z-index → deț ee elem.

apari în felică și spate

- 0 → în fundal

- n codă cu prioritate

- relative → punem elem cu position absolut

⇒ calc. distanța din div, nu de la marginile paginii

More CSS Features

- Media Query repeat pt min-width
↑ ne face pt dif dimensiuni
@ media (max-width: 600px) {
 - .video-grid {
grid-template-columns: 1fr 1fr;
}}

altă formă: @ media (min-width: 700px) and (max-width: 3000px) {}

CSS Shorthand Properties

- padding: 4px 10px 20px 30px; dacă am doar 2 val. (vertical, horizontal)
 - ↑ top
 - ↑ right
 - ↑ bottom
 - ↑ left
- margin
- border: 1px solid black;
 - ↑ style

Semantic Elements (same as div)

- nav
 - header
 - main → main content
 - ↳ section (în loc de div)
- <! -- -->
comment

Sprite-un CSS

- = o colecție de imagini a cărora se face la formă și se utilizează

HTTP(s) Protocol

(HyperText Transfer Protocol)

- for viewing web pages
- info sent in clear text
- vulnerable \Rightarrow HTTPS
↓
secure
- = este http dar cu security feature
↓
encrypt data

- protocol stateless = fiecare cerere de la client este tratată independent, fără a ţine cont de cererile anterioare
 - nu păstrează info despre istoricul utilizator

AVANTAJE

- simplificare (reducere gestionarea de pe server)
- scalabilitate (serverul poate suporta mai multe cereri)
- fiabilitate (reducere complicație)

DEZAVANTAJE

- transm. date suplimentare
- complexitate pe client

SSL

(Security Sockets Layer)

- protocol de securitate
- folosește public key encryption
- practic se face identificarea siteului prin intermediul certificatelor

TLS

(Transport Layer Security)

- succesorul SSL
- cel mai bun standard criptografic
- autentifică serverul și clientul și cripteză datele

Metodele Get și Post

- GET : solicita date de la un server (param. atașati URL-ului) + lim. de lungime + caching
- POST : trimite date către server (inclusiv în corpul cererii)

Cookies

- = mici fișiere stocate pe computer de către browser la solicitarea unui site web
- stocăază info despre sesiune, preferințe

1. Cookies de sesiune

- temporare
- șterse la inchidere browser

3. Cookies proprii (first cookie)

- gestionare conturi

2. cookie-uri persistentă

- persistente pe o perioadă fixă
până la stergerea manuală

4. cookies terți (third-party cookies)

- ex: Google AdSense

Java Script

- interpretat limbă de lime (nu ea compilor)

- JS Engine → transp codul JS în cod machine → faster

- <script src = "app.js"> </script>

ex: console.log('Ei x');

- let m; → am declarat o variabilă

→ int, string, etc

- m = new Object(); oice var eare nu e primitivă,
va crea niște obiect

- dacă nu adăugăm ;, se va adăuga singur

- const x = 10; nu poate fi modif.

- var b = 'a aa'; variabilă

- function add() { → poate returna,
} poate avea parametri

- pot fi folosite și ca const add = function a () {

return -

}

- var: scop global | fct

- let: scop de bloc

- function Higher Order (fun)


```
fun() {
    return function() {
        }
    }
}
```

- someX (num, obj)


```
↓   ↓ o refineră
a copy
```

- Object . getPrototypeOf (Human);

- pot face clase

```
class Human {
```

```
constructor (name) {
```

```
};
```

} → pot face metode globale prin eur static

- array, set, map, WeakMap

! garbage collection = dealoca
automat obiecte din memorie

Non-Blocking Event loop

- gestionarea operațiilor asincrone (mai multe conexiuni simultan)

- un acces la un singur thread ⇒ set Timeout

- se folosesc Promise → obiect ce reprezintă succesul sau eșecul unei op. asincrone


```
↑
evită callback hell
```

- addEventListener

- document . querySelector ('.button')


```
↑
css code
```

DOM
(Document Object Model)

- const person = h


```
aaa : () => h
console.log (this)
}
```

- const human = h

dma: 'AACT'

name: 'Jeff'

born -

properties

}

prototip = mecanism care permite
obiectelor să mențină
proprietăți și metode ale
alte obiect

Callback hell = utilizare excesivă

o: imbricătă a callbackelor

DOM = o interfață între contineuri

d) structura pag. web

- `typeof` → folosit pt a det. tipul unei variabile sau expr.

d.e.x: `console.log(typeof 3)`; se va afișa `number`

- `eval("2+2")` ⇒ 4.

↳ pericolosă pt că poate exec. pt decursive de cod

ECMA Script

- De bazat pe JavaScript

- astăzi o speciație standardizată pe care dif. implementări să o urmăreze

- every browser have JS engine

Node.js

- un mediu de execuție pt JS. construit pe V8 (Google Chrome)

- JS nu mai este limitat doar la utilizarea în browser

- folosit de diverse web

Primitive / Value Types

- String

- Number

- Boolean

- undefined

- null

Reference Types

- Object → ca o clasă din Domain

- Array

- Function

`new Array(1, '10', --, {x: 5, y: 5});`



poate fi inclusiv

tablouri în tablou

↑
Object

! dacă apelați o funcție cu mai mulți param.,
iar eu nu îți setez o parte dintr-unul,
se vor seta automat ca `undefined`

API JS = o serie de opere
pt Array

Operatiile pe Array - unu

- push, pop
- shift () → sterge și return - primul elem.
- unshift (elem) → inserarea la începutul arrayului și return lungimea
- slice (pos, m) → subsecție de la pos cu nr. de elem
- splice (pos, m, list) → stergere de la pos m. de elem.
 ↑
 o
 nu se sterg elem. ⇒ se inseră
 punte în list elem stăru
- indexOf (elem), lastIndexOf (elem)
- with (obj) ↳
 - pot accesa div. prop. fără
 - o repetă obj. - -

DESCURAJAT !!

! dacă am creat deja

obiectul, pot adăuga pe
parcurs atribută

ex. person. abc = false;

- == vs. ===

↑
verif că dă un rezultat
același tip

Obiectul window → abstracția fereastra browserului

- window.alert ('DA') → poate fi folosit și altfel simplu
- document (HTML)
 ↳ reprezintă pag. web encodată
 - proprietăți: title, URL, domain, referrer, body, head etc.

JSON = format neșarrit pt
(js obj notation) schimbul de info între
server și aplicații web

BOM = colecție de obiect pt
manipulare și interacțuirea
browserului web din JS

DOM și manipularea DOM

- o structură jerarhică de obiecte construită de către browser
- pt a facilita manipularea paginii

Evenimente

- proprietate elem (metode/membri) - fixate oferite cu prefixul dom
- pot utiliza this pt a face referire la elem. HTML care a declarat un eveniment
- event.target → face referire la obiectul din DOM care a declanșat evenimentul
- element.addEventListener → ascultă evenimentele pe un elem și exec. fct parcurgătoare

Timeout

- setTimeout → apel fct după x milsec. (codul JS se rulăză dintr-o bucată)
- pt ca să se răspundă la evenimente care durează mai mult decât x milsec, adăugăm setTimeout la finalul evenimentului
- clearTimeout → anulează setTimeout
- setInterval / clearInterval → apelăză periodic o părțile de cod deosebite de setTimeout

Încărcarea dinamică a unui fișier JS

- ulterior încărcării paginii să se exec. codul JS

<script>

```
var newScript = document.createElement('script');
newScript.src = 'extern.js';
document.body.appendChild(newScript);
```

</script>

Atenție la:

- apel script JS dinăuntru în tagul body
- evenimentul imaginativ se face doar când se încarcă în DOM

jQuery

- = JS framework
- Simplifică JS

exemplu `$(selector) = .query Select()`

- putem anima chestii
- pt a putea scrie în jQuery, trebuie să adaugi "pluginul" de pe siteul oficial sau sunsa sănii se afle în sistem
- document.getElementById("test").onclick = function()
 {
 alert('---');
 }

```
$("test").click(function()  
{  
    alert("___");  
});
```

în jQuery

`$(document).ready(function() {`

`});`

→ se rulează după ce domeniu încărcă

- `$(this).click();`

Events

- hover
- dblclick
- click
- keypress
- mouseout

- toggle → ascunde/afisă elem., și pot adăuga și întârziere

- fadeIn → ca toggle doar

- fadeOut efect

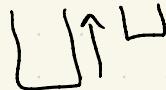
- fadeToggle

- fadeTo (1000, 0.5) → il face semi-transparent
 ↑
 transparență

CDN = rețea de servere distribuite
a conținutului afișată la
un acțor meniu pe aceneia
www

Slide effect

- slide Up (1000);
- slide Down
- slide Toggle



în loc de multi `sec` pot adăuga "slow"

Animation function

```
$( "#.nameclass" ). click (function () {
```

```
    $( "#.nameclass" ). animate( );
```

```
});
```



! pot elimina efectele animatiei cu
încă un `animate` în continuare pentru
ca să revină la val inițial CSS

width: '500px', → cod CSS
height: '300px'

```
}, 1000
```



dacă vr. să adauge
în timp

- dacă vreau să opresc animația, mai pe mă
un buton care va apela `stop`

Animated scroll

```
$( "html, body" ). animate( {
```

scrollTop: \$("#scroll - " + page - id) . offset . top,

```
}, 1000);
```

substituim (x) → preia de la

pe x încolo

dacă sun
- doar va lăsa

20px sus

View password input

- am o variabilă booleană care să mi se să fie să fie să nu
- setez atributul `type` la `text`

- `$("#text1"). text()` → preia textul (DOAR)

↳ și `$("input"). text()`
pot scrie `$("input"). text()`

- fol. `append (- -)`; pt a adăuga

alternative

- prepend (în conținut)
- before (outside)
- after (outside)

AJAX

(Asynchronous JavaScript on XML)

- nu-ți trimite toată pag. cîdoră lunc. dim ea reacșone
↳ practic iau date din DB și să dau refresh la tot website-ului
- ex: show more
 - o pot include la un button
 - \$("#test"). load("data.txt", { Name: "Danice", --- }, function() {
 - modify im. txt file
 - ↓
 - alert();
- pt a putea încredea date din DB, va trebui să folosim PHP
 - pt a face conexiunea
 - callback
- practic la primul load vor da Ligand php
- ca să afișez limitat, voi avea o var care contează de la 0

Get method

```
$("button"). click(function () {  
    $.get("data.txt", function(data, status) {  
        $(" #test" ). html(data);  
        alert(status);  
    })
```

Post method

```
$("input"). keyup(function () {  
    var name = $("input"). val();  
    $.post("file.php", { Name: name })  
})
```

Validare formular

`$("form").submit (function (event) {`

`event.preventDefault();`

`// extract data from elements`

`$.("form - msg").load("mail.php",`

`↳ Name : name,`

{

}

Validation se face
la nivel de cod
PHP

PHP : HyperText Pre processor

`<?php --- ?>`

- `echo ("Hello");`
- `$ hello = '';` → variabili
- close function
- function anonymous: `array_map(function($a,$b){`

`return $a * $b;`

})

- one DOP → pot face close

- se puntează în HTML file

- dacă scriu în echo HTML, it works like html code.

`echo "<h1> JA </h1>"`

Afișare cu var: `echo "Age: $age";`

Data Types

- string
- integer
- decimal numbers
- boolean
- null value

String Fct

- `strtolower ($str);`
- `strtoupper`
- `strlen`
- `$str[0] → acces la lit de poz. 0.`
- `str_replace (string de înloc., înlocuitor, variabila)`
- `strstr (von, x, y) căk`

- la numere se comportă ca în C++, inclusiv `++`, `--`, `+=` cătă și fără dim ceea ce
 - pt a obține user input: `$ GET ["name"]`
- ↑
numere din HTML
- Văzută dintr-un form**
se dă prim URL
(de văz acolo !!)
- vulnerabilitate = password \Rightarrow SOLUȚIE:
POST în loc de
`GET`
- ex: localhost:4000/file.php?name=David

```
$ array = array( "A", "B", "C");
```

- nu-l pot afisa direct (trebuie să fie un param: array[0])
- pot schimba oricărui tipul unei date memorate: array[1]=400
- count function
- adaug astfel: array[4]= "D"
- variantele selectate la checkbox' se pun în array
- associative arrays (like map in java) ex: array ("A"=>"B", "C"=>"D")
- if (--) {
 } else {
 } sau elseif (--) {
 }
 }
- <? php include "header.html" ?>

Obiecte

```
class Bee {
  $obj = new Bee; anum. cu $obj = new Bee();
  public var $aa;
  private var $b;
}
```

$\$obj \rightarrow aa = "hei";$
 $\$obj \rightarrow b = "DA";$

pot crea fct - construct() \rightarrow ca în java

- pot avea clasă în clasa \Rightarrow clasa internă

↳ nu poate fi folosită direct, ci primul felul de instanțare în clasa principală

Web Security

SQL Injection

- atacatorul inseră cod malitios în DB pt a obține date
- trebuie făcută validare (inclusiv îmbunătățiri cod SQL)
- paronul se hashăază (nu e criptat!)
- dacă fac validare pe frontend, tot trebuie să fac și pe backend

XSS Attack

(Cross - Site Scripting)

- atacatorul introduce comentarii malicioase în aplicația web.
- 3 tipuri: persistente, non-persistent, DOM based
- realizat prin intermediul paron. trimitând spre backend părți GET / POST
add comm: `<script>` -- `</script>`
- SOL: validare pe backend

Cross - site Request Forgery Attack (CSRF)

- atac major asupra aplicației web
- atacatorul execută acțiuni mediate pt a păcăli userul
- în URL acel get / post
- SOL: tokeni asențiali (generare chiar)

Unrestricted File Upload

- se încarcă alte tipuri de fișiere (ex. .php)
- SOL: verificare extensie pe backend.

Path Traversal

- ex: atacatorul poate invoca un fișier de ex: delete.php prin URL

SMTP headers injection

- validează că la fișier nu fie doar emailul bune
funcție: filter-var

Tips + Arte cheati

- coduri de eroare cu doar cifre în B16
- coduri 3xx → coduri de redirect. în HTTP

! nu toate tagurile acceptă atributul name

Un formular care contine un input de tip file trebuie:

Select one or more:

sa aiba specificat atributul enctype setat la multipart/form-data

- • op de concatenare în PHP
- dacă am .info.member → cauță tagul ce le conține pe ambele

Stări AJAX

- 0 - UNSENT
- 1 - OPENED
- 2 - HEADERS_RECEIVED
- 3 - LOADING
- 4 - DONE

<link> conectare CSS file

× SS - extensiișe funcț.