Suport Curs TW

**CURS 1**

***WWW – WORLD WIDE WEB***

• 1957 – URSS, Sputnik

• 1962 – SUA, Advanced Research Projects Agency

• 1980 – rețeaua ARPAnet devine Internet, utilizând TCP/IP

• Email – 1984, https://www.youtube.com/watch?v=szdbKz5CyhA

Internet Shopping – 1984, https://www.youtube.com/watch?v=ZsV1e1XCmWk

• 1990 – 12 noiembrie, CERN (Conseil européen pour la recherche nucléaire, Geneva)

• 1991 – HyperText Transfer Protocol (HTTP)

***HTTP – HYPERTEXT TRANSFER PROTOCOL***

• - mecanism fără stare, bazat pe o comunicare de tip cerere-răspuns, prin care sunt gestionate sistemele

informatice hypermedia

• 1991 – HTTP 0.9

• 1996 – HTTP 1.0, https://tools.ietf.org/html/rfc1945

• 1997 – HTTP 1.1, https://tools.ietf.org/html/rfc2068

• 2015 – HTTP 2.0, https://tools.ietf.org/html/rfc7540

• 2018 – HTTP 3.0, https://quicwg.org/base-drafts/draft-ietf-quic-http.html

***WEB SERVER***

• 1990 – CERN httpd, Tim Berners-Lee

• 1993 – NCSA httpd, Robert McCool (National Center for Supercomputing

Applications), a introdus CGI (Common Gateway Interface)

• 1995 – Apache (2.4.43, https://httpd.apache.org/)

• 1995 – Internet Information Services (IIS 10, https://www.iis.net/)

• 2003 – LiteSpeedWeb Server (LSWS 5.4.1, https://www.litespeedtech.com/)

• 2004 – Nginx (1.16.1, <https://nginx.org/>)

***CLIENT - WEB BROWSER***

• 1990 – WorldWideWeb, Tim Berners-Lee

• 1991 – Line Mode Browser, Nicola Pellow (CERN)

• 1993 – Mosaic > Netscape (1994-1998) > Mozilla Firefox (77, 2004)

• 1995 – Internet Explorer (11, Microsoft) a devenit Microsoft Edge

• 1996 – Opera (62)

• 2003 – Safari (Apple)

• 2008 – Chrome (81, Google)

***HYPERTEXT MARKUP LANGUAGE (HTML)***

• 1991 – versiunea 1, Tim Berners-Lee (1989)

• 1995 – versiunea 2 (1993) introduce form, https://tools.ietf.org/html/rfc1866

• 1997 – versiunea 3 (1995), https://www.w3.org/TR/2018/SPSD-html32-20180315/

• 1998 – versiunea 4, https://www.w3.org/TR/html40/

• 2002 – XHTML 1 (2000), https://www.w3.org/TR/xhtml1/

• 2014 – HTML 5 (2008), https://tools.ietf.org/html/rfc7992

Oficial: https://www.w3.org/

Wiki: <https://www.w3.org/History/19921103-hypertext/hypertext/WWW/History.html>

***CASCADING STYLE SHEETS (CSS)***

• 1996 – W3C, https://www.w3.org/TR/REC-CSS1-961217

• 1998 – https://tools.ietf.org/html/rfc2318

• Site oficial: <https://www.w3.org/Style/CSS/>

***JAVASCRIPT (JS)***

• 1995 – lansarea prin Netscape

• 1995 – JScript (Microsoft, reverse engineering)

• 2017 – JSON, https://www.ecma-international.org/publications/standards/Ecma-404.htm

• 2019 – ES 6, https://www.ecma-international.org/publications/standards/Ecma-262.htm

• 2009 – Node.js – web framework (13)

• Site oficial: https://www.ecma-international.org

• Youtube: <https://www.youtube.com/playlist?list=PL62E185BB8577B63D>

***TEHNOLOGII WEB***

• HTML5, CSS, Javascript

• HTTP: cerere/răspuns, URL, headers, querystring, content-type

• ResponsiveWeb Design: LESS/SASS, Bootstrap

• Cookies, localStorage, sessionStorage

• XML, JSON, AJAX, Webservices

***UNELTE***

• Editoare de text: Notepad, Notepad++, Sublime Text, Atom

• IDE: Visual Studio Community Edition, Visual Studio Code

• Versionare/partajare (git): GitHub, GitLab, BitBucket

• Virtualizare: Docker (containere), VMWare Player, VirtualBox

• Tranzacții HTTP: Postman, browsere (Google Chrome, Mozilla Firefox, Microsoft Edge)

***DOCKER***

• https://www.docker.com/products/docker-desktop

• Docker Hub: https://hub.docker.com/

• Crearea unui server web (Apache) pentru o aplicație web statică (doar cu HTML5):

- pentru Windows OS:

docker run --name my-apache-static -d -p 83:80 -v ${pwd}:/usr/local/apache2/htdocs/ httpd:2.4

- pentru Linux/MacOS:

docker run --name my-apache-static -d -p 83:80 -v $(pwd):/usr/local/apache2/htdocs/ httpd:2.4

docker run --name my-apache-static -d -p 83:80 -v “$(PWD)”:/usr/local/apache2/htdocs/ httpd:2.4

**CURS 2**

***ANATOMIA APLICAȚIILOR WEB***

Una dintre cheile comunicării este locul în care se află o resursă web. Identificarea se face unic cu ajutorul URL (Uniform Resource Locator sau Uniform Resource Identifier, adică URI).

Sintaxa unui URL este:

<schema>:<authority>:<port>/<resursa>?<interogare>#<fragment>

<schema> <authority> <port> <resursa> <interogare> <fragment>

<schema> este protocolul utilizat: http, https, ftp, ssh, smtp, etc.

<authority> poate fi de forma unei adrese IP sau a unui nume de domeniu, dar precedat de // în cazul http şi https, spre exemplu.

<port> este numărul portului utilizat, dar poate să lipsească dacă este folosit portul implicit al protocolului. Spre exemplu, pentru http portul este 80, pentru https este 443, pentru ftp este 21, pentru ssh este 22, iar pentru smtp este 25.

<resursa> poate fi o pagină statică HTML, PDF, alt tip de fişier sau o resursă dinamică (generată dinamic de server).

<interogare> este de forma <cheiel>=<valoarel> sau <cheiel>=<valoare1>&<cheie2>=<valoare2>

<fragment> poate fi valoarea atributului id al unui element HTML din resursă.

***URL***

sintaxa pentru localizarea și accesarea resurselor în Internet

https://www.rfc-editor.org/rfc/rfc1738

derivat din specificațiile URI (Uniform Resource Identifiers), https://www.rfc- editor.org/rfc/rfc1630

• sintaxa pentru web: http://<host>[:<port>]/<path>[?<searchpart>][#<fragment>]

codificarea caracterelor în URL (https://tools.ietf.org/html/rfc3986#section-2.1) se face prin %HH, unde HH este codul ASCII exprimat hexazecimal

***TRANZACŢII HTTP (1)***

Comunicarea cu ajutorul protocolului HTTP se realizează urmând paşii:

1. Se redactează un URL în interfața unui client (de exemplu, browser)

2. Clientul trimite o cererea HTTP către un server web

3. Serverul web identifică resursa solicitată sub forma unui fișier sau agreghează datele solicitate printr-o aplicație software

4. Serverul web returnează un răspuns HTTP

5. Clientul (browser-ul) asamblează și afișează răspunsul primit

***CERERI HTTP***

Pentru a realiza operații CRUD (Create-Read-Update-Delete) cu resursele hypertext prin tranzacții HTTP se pot folosi următoarele tipuri de cereri:

POST (Create, non-cacheable)

GET (Read, cacheable)

PUT (Update, idempotenta)

DELETE (Delete)

***CEREREA HTTP***

Sintaxa unei cereri HTTP contine:

-o linie cu tipul cererii, calea şi versiunea cererii (separate prin spații)

- antetele cererii (listă de parametri cu valori asociate)

- linie vidă

-corpul cererii(optional)

***RĂSPUNSUL HTTP***

După ce serverul procesează cererea HTTP, returnează un mesaj de răspuns HTTP compus din:

-linia de status

-antetele de răspuns

-linie vidă

- corpul mesajului (optional)

***RĂSPUNSUL HTTP***

-Linia de status conține:

-versiunea de HTTP

-codul status: 1xx, 2xx, 3xx, 4xx, 5xx

-o descriere scurtă a codului

Exemple:

HTTP/1.1 200 OK

HTTP/2.0 404 Not Found

HTTP/1.1 500 Internal Server Error

Detalii: https://httpwg.org/specs/rfc9110.html#overview.of.status.codes

Antetele răspunsului sunt formate din perechi de forma cheie/valoare:

-Content-Type: text/css

-Content-Length: 40

-Cache-Control: no-cache

***TESTAREA TRANZACȚIILOR HTTP***

Testarea tranzacțiilor HTTP se poate face, în plus, cu excepția browser-elor şi cu programe utilitare, precum:

-telnet

-Terminal

-curl

-Postman

***STRUCTURA UNUI DOCUMENT HTML5***

<!DOCTYPE html>

<html lang="ro">

<head>

<meta charset="utf-8" /> <title>Titlul paginii</title>

</head>

<body>

</body>

</html>

***MARKUP - ELEMENT HTML***

https://html.spec.whatwg.org/multipage/dom.html

<marcaj atribut

valoare

eveniment

actiune();" />

<marcaj atribut valoare eveniment actiune();

Container/Text

</marcaj>

***HYPERTEXT MARKUP LANGUAGE (HTML)***

https://html.spec.whatwg.org/multipage/dom.html

• Categorii:

• Metadata

• Flow:

• Heading

• Sectioning

• Phrasing

Embedded

Interactive

***HTML - Metadata***

<title>Titlul documentului</title>

<base href="https://photos.mysite.ro" target="\_blank" />

• href: URL

target: self (default), blank, parent, \_top

<meta charset="utf-8" />

<meta name="keywords" content="bicicleta, piese, anvelope" />

***HTML - Sectioning***

<article> ... </article>

<section>> ... </section>

<aside>...</aside>

<nav> ... </nav>

***HTML Heading -***

<h1>Titlul cel mai mare</h1>

<hgroup>

<h2>Titlul mare</h2>

<h1>Titlu</h1>

<h3>Titlu</h3>

<h2>Subtitlu</h2>

<h4>Sub-titlu</h4>

</hgroup>

<h5>Sub-sub-titlu</h5>

<h6>Titlul cel mai mic</h6>

***HTML - Flow***

<header></header>

<nav></nav>

<main></main>

<aside></aside>

<footer></footer>

<!-- comentariu HTML -->

***HTML - Flow***

<ul>

<li>Item 1</li>

<li>Item 2</li>

</ul>

<ol>

<li>Item 1</li>

<li>Item 2</li>

</ol>

<dt>Termen</dt> <dd>Definiţie</dd>

</dl>

<nav>

<ul>

<li>Item 1</li>

<li>Item 2</li>

</ul>

</nav>

**CURS 3**

**URL**

-sintaxa pentru localizarea și accesarea resurselor în Internet

https://www.rfc-editor.org/rfc/rfc1738

-derivat din specificațiile URI (Uniform Resource Identifiers), https://www.rfc- editor.org/rfc/rfc1630

-sintaxa pentru web: http://<host>[:<port>]/<path>[?<searchpart>][#<fragment>]

-codificarea caracterelor în URL (https://tools.ietf.org/html/rfc3986#section-2.1) se face prin %HH, unde HH este codul ASCII exprimat hexazecimal

***ELEMENTUL HTML TABLE***

Este folosit pentru stilizare (!) sau afişarea datelor tabulare.

• Container: <table>...</table>

Containere copii: <caption>...</caption>, <tr>...</tr>,

<thead>...</thead>, <tbody>...</tbody>, <tfoot>...</tfoot>

În <thead>...</thead> se vor folosi linii pentru afișarea antetelor: <tr>...</tr>

În <tbody>...</tbody> se vor folosi linii pentru afişarea datelor: <tr>...</tr>

În <tr>...</tr> se vor folosi <th>Antet</th> sau <td>Date</td>

**ELEMENTUL HTML FORM**

A apărut în 1993 cu scopul de a trimite date spre serverul web!

Container: <form>...</form>

• Atribute importante:

method="get|post"

• action="URL"

enctype="application/x-www-form-urlencoded"

Etichete (optional): <label for="numeID">Introduceți numele:</label>

Text: <input id="numeID" type="text" name="nume" />

***FORMULARE HTML - MARCAJE I***

Text: <input type="text" name="nume" />

Text mascat: <input type="password" name="parola" />

Text ascuns (vizual): <input type="hidden" name="pasintermediar"

• Imagine: <input type="image" name="pozitieinimagine" />

Text (căutare): <input type="search" name="cautare" />

Text multilinie: <textarea name="descriere" />

• Buton: <input type="button" value="Actualizează" />

Submit: <input type="submit" value="Trimite" />

Resetare: <input type="reset" value="Resetare" />

***FORMULARE HTML - MARCAJE II***

• Opţiuni (care se exclud; gruparea se face cu aceeaşi valoare pentru name):

<input type="radio" name="tipreteta" value="pizza rustica" />

<input type="radio" name="tipreteta" value="pizza diavola" />

• Opţiuni (care se reunesc; acelaşi name va conduce la un tablou de valori):

<input type="checkbox" name="ingredient" value="brânză" />

<input type="checkbox" name="ingredient" value="dovlecei" />

<input type="checkbox" name="livrare" value="da" />

***FORMULARE HTML - MARCAJE III***

Opţiuni (care se exclud):

<select size="4" name="tipreteta">

<option value="Pizza Rustica" selected />

<option value="Pizza Diavola" /></select>

Opţiuni (care se reunesc):

<select multiple size="4" name="ingredient">

<option value="ciuperci" selected />

<option value="dovlecei" /></select>

***FORMULARE HTML - MARCAJE NOI I***

Buton: <button>Trimitere</button>

Email: <input type="email" name="email" />

Dată: <input type="date" name="data" />

Timp: <input type="time" name="timp" />

Dată şi timp: <input type="datetime-local" name="datasitimp" />

Input/Select: <input list="Optiuni" name="optiune" />

<datalist id="Optiuni">

<option value="Pizza Rustica" />

<option value="Pizza Diavola" />

</datalist>

***FORMULARE HTML - MARCAJE NOI II***

> Culoare: <input type="color" name="culoare" />

> Interval de valori: <input type="number" name="cantitate" min="1" max="5" />

> Interval de valori: <input type="range" name="cantitate" min="1" max="5" />

Telefon: <input type="tel" name="telefon" pattern="[0]{1}-[7]{1}-[0- 9]{8}" />

Url: <input type="url" name="urlwebsite" />

♦ Săptămâna: <input type="week" name="saptamana" />

***FORMULARE HTML - ATRIBUTE***

→ Atribute cu valoare:

formaction, formmethod, formenctype

name, value, id, for

rows, columns, maxlength

type, size, placeholder, min, max, step

→ Atribute booleene:

✩ required, disabled, readonly, multiple

**CURS 4**

***CSS = CASCADING STYLE SHEETS***

Foi de stil în cascadă

Definirea aspectului şi prezentarea documentelor HTML şi XML(!)

1996 - W3C, https://www.w3.org/TR/REC-CSS1-961217

1998- https://tools.ietf.org/html/rfc2318

Site oficial: https://www.w3.org/Style/CSS/

♦HTTP Response Header: content-type: text/css

***CSS - INTEGRARE***

Browser: prin setări sau extensii

> Extern: prin marcajul link, specificând URL-ul unei resurse de tip CSS

<link href="URL" rel="stylesheet" media="all/print/screen" />

> Intern: marcajul style, atributul class sau atributul style.

<style media="all/print/screen">

/\* instrucțiuni CSS \*/

</style>

<tag class="clasamea" style="instructiuniCSS; />

***SINTAXA CSS***

selector{ proprietate: valoare; }

***VALORI CSS***

→ Valori numerice absolute: px, cm, mm etc

→ Valori numerice relative: em, rem, vw, vh etc

Culori (#RRGGBB), #3fabd2, #aef, blue, lightgreen etc

Stil: none, solid, dotted, dashed etc

Text: content: "→"

Funcţii: url("myimage.jpg"), rgb(r,g,b), rgba(r,g,b,a), calc(100% 80px) etc

***SELECTORI CSS***

♦ universal:

\* { ... }

> element:

p { ... }

class:

Id:

#myid { ... }

♦ grup:

h1, h2, h3 { ... }

Combinate:

.clasamea { ... }

p.clasamea {}

***VARIABILE CSS***

:root {

--r: 240;

--g: 235;

--b: 0;

--w: 100px;

-- culoare: rgb(var (--r), var (--g), var(--b));

}

div{

width: var(--w);}

***MODELUL BOX***

→ Proprietăţi CSS definibile:

> width: lăţimea (1 dimensiune)

> height: înălţimea (1 dimensiune)

◇ padding: umplutura (4 dimensiuni)

border: frontiera (4 dimensiuni-stil-culoare)

margin: exteriorul (4 dimensiuni)

***CSS - ALŢI SELECTORI***

Combinatori: div>p

Pseudo-clase: a:hover

Pseudo-elemente: h1: :before

Selectori cu atribute: p[width~="100%"]

**CURS 5**

***Web Design***

<table

display: block; /\* inline-block \*/

float: left;

display: flex; /\* inline-flex \*/

display: grid; /\* inline-grid \*/

***WEB DESIGN - TABLE***

→ Structura este definită cu ajutorul elementului HTML table, dar stilizată şi dimensionată

prin CSS

Greu de implementat pentru că are mult atât HTML cât şi CSS

***WEB DESIGN - DISPLAY: BLOCK***

→ Necesită cunoştinţe avansate de CSS

→ Elementele HTML sunt afișabile în format block sau inline, dar se pot schimba prin CSS

> Implicit, elementele de tip block se afişează pe câte o linie, iar cele de tip inline se afişează unul după celălalt pe aceeaşi linie şi se continuă pe următoarea

Setarea dimensiunilor (width şi height) este vizibilă doar pentru cele de tip block

◆ Pentru ca dimensiunea unui element să fie calculată inclusiv după frontieră (border) trebuie declarată la nivelul său box-sizing: border-box;

***Web design - display: block***

→ Elemente de tip block:

<address>, <article>, <aside>,<blockquote>,<canvas>,<dd>,<div>,<dl>,<dt>,<fieldset>, <figcaption>,<figure>, <footer>, <form>, <h1>-<h6>, <header>, <hr>,<li>,<main>,<nav>, <noscript>,<ol>,<output>,<p>,<pre>,<section>,<table>,<tfoot>,<ul>,<video>

→ Elemente de tip inline:

<a>, <abbr>,<b>, <bdi>,<bdo>,<br>,<button>,<cite>, <code>,<data>,<datalist>,<del>,<dfn>, <em>,<i>,<iframe>,<img>,<input>,<ins>, <kbd>,<label>,<map>, <mark>, <meter>, <noscript>, <object>,<output>,<picture>,<progress>, <q>,<ruby>, <s>,<samp>,<script>,<select>, <small>, <span>,<strong>,<sub>,<sup>,<svg>,<template>,<textarea>,<time>,<u>,<var>,<wbr>

***Web design - display: block***

→ Design-ul se implementează prin: display: inline-block;

→ Elementele vor fi afişate în blocuri conform dimensiunilor specificate şi vor continua pe linia următoare

→ Pentru a fi lipite au apărut mai multe variante, precum:

.clearfix:after {

content:"";

display: table;

clear: both;

***Web design - float: left***

Se implementează prin: float: left;

Pentru a anula,,lipirea" elementelor trebuie implementată clear: both;

→ Poziționarea se face cu ajutorul proprietăților position, left, right, top, bottom

Elementele vor fi afişate în blocuri şi vor fi lipite la stânga (sau la dreapta, după caz)

> Pentru a fi lipite au apărut mai multe variante, precum:

.clearfix:after {

content:"";

display: table;

clear: both;

}

***WEB DESIGN - DISPLAY: FLEX***

→ Structura HTML este de tipul: Container care conţine elemente de tip block

→ Containerul este tip Flex, dacă este definit prin: display: flex

→ Doar nodurile copil (de tip block) sunt elemente Flex

◆ Mai multe containere Flex vor fi afişate inline prin: display: inline-flex

→ Elementele Flex sunt afişate pe linie sau pe coloană, în funcţie de modul în care au fost definite prin proprietatea flex-direction

***ELEMENTE FLEX - ORIENTARE***

column-reverse

Pot fi aranjate pe orizontală (implicit, de la stânga la dreapta) sau pe verticală:

flex-direction: row; /\* implicit \*/

flex-direction: column;

flex-direction: row-reverse;

flex-direction: column-reverse;

***ELEMENTE FLEX - ÎNCADRARE***

Implicit, elementele sunt afişate pe o singură linie. Prin setarea valorii wrap elementele se pot afişa pe mai multe linii:

flex-wrap: nowrap; /\* implicit \*/

flex-wrap: wrap;

flex-wrap: wrap-reverse;

Proprietăţile flex-wrap şi flex-direction se pot seta împreună cu ajutorul proprietăţii

flex-flow:

flex-flow: column wrap;

***ELEMENTE FLEX - DIMENSIONARE***

> Implicit, elementele sunt dimensionate auto (adică în funcție de conţinut), dar se poate seta cu ajutorul proprietăţii flex-basis. Alte proprietăţi care ajută la redimensionarea elementelor sunt flex-grow şi flex-shrink.

flex-grow: 0; /\* implicit \*/

flex-shrink: 1; /\* implicit \*/

flex-basis: auto; /\* implicit \*/

flex: 01 auto; /\* flex-grow flex-shrink flex-basis \*/

***ELEMENTE FLEX - ALINIERE***

Implicit, elementele sunt aliniate la stânga (pentru că de la stânga se afişează pe orizontală). Alinierea se setează la nivelul Containerului Flex, astfel:

◊ pe orizontală prin justify-content, iar valorile posibile sunt flex-start, center, flex- end, space-between, space-around.

◊ pe verticală prin align-items, iar valorile posibile sunt flex-start, center, flex-end, baseline, stretch.

Mai multe detalii despre Flex: <https://www.w3.org/TR/css-flexbox-1/>

***WEB DESIGN - DISPLAY: GRID***

→ Structura HTML este de tipul: Container care conţine elemente de tip block

→ Containerul este tip Grid, dacă este definit prin: display: grid

→ Doar nodurile copil (de tip block) sunt elemente Grid

> Mai multe containere Grid vor fi afişate inline prin: display: inline-grid

→ Elementele Grid sunt afişate într-un sistem de tip Grid identificat prin linie şi coloane (drepte orizontale sau verticale numerotate de la 1) sau prin zone

***CONTAINERUL GRID***

→ Setări (obligatorii):

display: grid;

grid-template-columns: auto 100px 1fr repeat (2, 200px) 1fr; grid-template-rows: 50px 1fr repeat (2, 200px);

grid-template-areas:

"title title"

"nav nav"

"sidebar main";

gap: 10px;

***ELEMENTE GRID - POZIȚIONARE***

Poziționarea pe coloană:

grid-column-start: 1;

grid-column-end: 3;

grid-column: 1/3;

Poziționarea pe linie:

grid-row-start: 2;

grid-row-end: 4;

grid-row: 2/4;

***ELEMENTE GRID - DELIMITAREA PRIN NUME***

Setări pentru Containerul Grid:

display: grid;

grid-template-columns: repeat (2,

1fr);

grid-template-rows: [title-

start] 200px[nav-start] 100px[nav- end] auto [page-end];

Poziționarea meniului prin:

grid-row: nav;

Poziționarea titlului:

grid-row: title-start/nav-start;

***ELEMENTE GRID - IDENTIFICAREA PRIN ZONE***

→ Setări pentru Containerul Grid:

display: grid;

grid-template-columns: repeat (2, 1fr); grid-template-rows: 200px 100px auto;

grid-template-areas:

→ Identificarea elementului pentru titlu: .titlu { grid-area: title; }

Identificarea elementului pentru meniu: .meniu grid-area: nav; }

Identificarea elementului pentru sidebar: .lateral grid-area: sidebar; }

"title title"

"nav nav"

"sidebar main";

→ Identificarea elementului pentru conţinut: .continut grid-area: main; }

**CURS 6**

***SASS= SUPERSET DE CSS***

→ Permite crearea de CSS într-o manieră rapidă

→ Declaraţiile CSS sunt conservate

→ Extensie VS Code: Live Sass Compiler (ID: glenn2223.live-sass)

Site oficial: https://sass-lang.com/

→ Vom folosi sintaxa SCSS şi extensia \*.scss

***SASS - INTEGRARE***

Se implementează separat și este livrat în format CSS, în urma compilării

Se poate integra în cadrul aplicațiilor web şi se poate compila cu ajutorul extensiilor disponibile în mai toate IDE-urile

Deşi se poate folosi şi în browser putând fi compilat la nivelul clientului cu ajutorul limbajului Javascript, nu se recomandă utilizarea în această manieră deoarece browserul are nevoie doar de CSS!

> Dacă avem un fişier CSS, îi putem schimba extensia (din css în scss) şi putem continua cu fişierul scss deoarece în aplicația web vom integra rezultatul compilării, adică fișierul css.

***SINTAXA SCSS***

proprietate: valoare;

***FIŞIERE PARŢIALE***

Fişierele parţiale se salvează cu denumirea \_fisier.scss (prefixate cu simbolul \_) şi se integrează în alte fişiere SCSS cu ajutorul regulii @use sau @import.

→ Ele pot conţine doar variabile sau reguli de generare SCSS şi din acest motiv nu trebuie compilate separat.

Integrarea lor se va face printr-una dintre cele două variante:

@use "fisier";

@import "fisier";

◆ Analog, putem integrarea şi un fişier stil.css:

@use "stil";

@import "stil";

Important: @import va dispărea din SCSS la un moment dat!

***VARIABILE SCSS***

> Tipuri de date: Numbers (12, 100px etc), Strings (Helvetica, sans-serif etc), Colors (red, #ae23cd etc), Boolean (true sau false), null, List of values ([50px, 100px, 150px]), Maps (("background": lightgreen, "foreground": blue)).

> Variabilele sunt identificatori prefixaţi cu simbolul $:

$font-stack: Helvetica, sans-serif;

$primary-color: #333;

$standard-width: 100px; // acest comentariu SCSS nu apare în CSS /\* aceste comentarii SCSS se văd în fișierul CSS obținut prin compilare \*/

body {

font: $font-stack;

color: $primary-color;

}

div { width: $standard-width; /\*! dimensiune standard, apare chiar și în CSS comprimat \*/

***FUNCŢII SCSS***

Se pot declara în orice fişier SCSS cu ajutorul regulii @function: @function pow($base, $exponent) {

$result: 1;

@for $ from 1 through $exponent {

$result: $result \* $base;

}

@return $result;

Apelul va fi de forma pow(4,3) dacă avem nevoie de valoarea 64-4³

→ Dacă putem avea un număr variabil de parametri:

@function sum ($numbers...) {

$sum: 0;

@each $number in $numbers {

$sum: $sum + $number;

}

@return $sum;

}

Apelul va fi de forma sum(4,3) dacă avem nevoie de 4+3=7 sau sum(30px, 20px, 50px) dacă avem nevoie de 30px+20px+50px=100px

***MIXIN***

Se pot declara în orice fişier SCSS cu ajutorul regulii @mixin şi se folosesc cu ajutorul regulii @include:

@mixin elipsa ($t1, $tr, $br, $b1, $bgcolor) {

border-radius: $tl $tr $br $b1;

background-color: $bgcolor;

}

div {

width: 100px;

height: 50px;

@include elipsa (5,5,10,10);

}

***MIXIN***

Într-un mixin putem avea şi parametri impliciti:

@mixin tema ($culoare: DarkGray) {

background: $theme;

box-shadow: 0 0 1px rgba($theme,

25);

color: #fff;

}

Exemple de implementări:

.info {

@include tema;

}

.alert {

@include tema ($culoare: DarkRed);

}

.success {

@include tema ($culoare: DarkGreen);

}

***EXTINDERI/MOŞTENIRI***

→ Definim un grup de declarații CSS pe care vrem să-l extindem prefixând un identificator cu %:

%mesaj-tip {

border: 1px solid #ccc;

padding: 10px;

color: #333;

}

// Nu se afişează dacă nu este implementat

%flex-orizontal {

display: flex;

flex-wrap: wrap;

}

→ Utilizăm o extindere cu @extend:

.mesaj {

@extend %tip-mesaj;

}

.success {

@extend %tip-mesaj;

border-color: green;

}

.error {

@extend %tip-mesaj;

border-color: red;

}

***LISTS ÎN SCSS***

Parcurgerea unei liste permite generarea mai multor declaraţii CSS:

$sizes: 40px, 50px, 80px;

@each $size in $sizes {

.icon-#{$size} {

font-size: $size/2;

height: $size;

width: $size;

}

CSS-ul generat va fi:

.icon-40px {

font-size: 20px; 40px; height: 40px; width:

}

.icon-50px {

font-size: 25px; height: 50px; width: 50px;

}

.icon-80px {

font-size: 40px; width: 80px; height: 80px;

}

***MAPS ÎN SCSS***

Parcurgerea unei hărți permite generarea mai multor declarații CSS folosind cheia şi valoarea:

$teme: ("primary": "gray", "success": "lightgreen", "danger": "red");

@each $denumire, $culoare in $teme { .bg-#{$denumire}:before { background-color: $culoare;

}

}

CSS-ul generat va fi:

.bg-primary: {

background-color: gray;

}

.bg-success: {

background-color: lightgreen;

}

.bg-danger: {

background-color: red;

}

***CULORI ÎN SCSS***

Culorile se pot modifica prin SCSS cu ajutorul mai multor funcții:

color.adjust (#6b717f, $red: 15); // #7a717f

color.adjust (#d2e1dd, $red: -10, $blue: 10); // #c8e1e7

color.adjust (#998099, $lightness: -30%, $alpha: -0.4); // rgba(71,

57, 71, 0.6)

Putem obţine şi alte informaţii despre culori:

color.red() // cantitatea de roşu color.green() // cantitatea de verde

color.blue() // cantitatea de albastru

color.hue() // nuanța

color.saturation() // saturația

color.lightness() // luminozitatea

***CULORI ÎN CSS PRIN HSL***

> H = este valoarea pentru nuanţa, valoare în grade, între 0 şi 360.

S = saturația culorii, valoare în procente, între 0 şi 100.

L = luminozitatea, valoare în procente, între 0 şi 100.

Dacă S = 0, atunci vom avea doar nuanţe de gri, indiferent de valoarea H, dar identificate prin valoarea luminozităţii L, de la culoarea albă (hsl(0,0,100)) şi până la negru (hsl(0,0,0)).

Red = rgb(255,0,0) = hsl(0,100,50), Yellow = rgb(255,255,0) = hsl(60,100,0)

Green = rgb(0,255,0) = hsl(120,100,50), Cyan = rgb(0,255,255) = hsl(180,100,50)

Blue = rgb(0,0,255) = hsl(240,100,50), Magenta = rgb(255,0,255) = hsl(300,100,50)

***MEDIA QUERIES***

→ Aplicarea declarațiilor CSS în funcție de mediul de afişare şi de anumiți parametri ai acestuia pentru a afişa acelaşi conţinut într-un design diferit

◆ Exemple în HTML şi, respectiv, în CSS:

<link href="URL" rel="stylesheet" media="screen and (min-width: 500px)" />

@media screen and (min-width: 500px) {

/\* blocuri de declarații CSS care se vor executa dacă sunt satisfăcute condițiile de ecran cu lățimea minimă de 500px \*/

Responsive Web Design

/\* declarații CSS pentru ecran cu lățimea maximà de 500px \*/

@media

screen and (min-width: 500px) {

/\* declarații CSS pentru ecran cu lățimea minimă de 500px \*/

}

@media screen and (min-width: 1000px) {

/\* declarații CSS pentru ecran cu lățimea minimà de 1000px \*/

}

/\* declarații CSS pentru ecran cu lățimea minimă de

1000px \*/

@media screen and (max-width: 1000px) {

/\* declarații CSS pentru ecran cu lățimea maximă de

1000px \*/

}

@media screen and (max-width: 500px) {

/\* declarații CSS pentru ecran cu lățimea maximă de

500px \*/

}

***CSS SPRITES***

→ Pentru a implementa procedeul CSS Sprites avem nevoie de un fişier imagine (de preferat în format png), care să conţină toate secvenţele pe care dorim să le implementăm şi de două clase CSS:

.setup {

background: url("imagine.png") repeat-none;

width: 100px; /\* lățimea secvenței \*/

height: 50px; /\* înălțimea secvenței \*/

.secventa-1 {

background-position: x y; /\* poziția colțului stânga-sus \*/

/\* x=orizontală, y=verticală (valori negative) \*/

}

***ECMASCRIPT = JAVASCRIPT =/ JAVA***

https://www.ecma-international.org/publications-and-standards/standards/ecma-262/

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference

Integrare în HTML:

Extern: prin marcajul script

> Intern: prin marcajul script sau prin evenimente, în orice element HTML

Limbajul Javascript foloseşte setul de caractere Unicode, este case-sensitive, iar instrucţiunile sunt interpretate de gazdă

***ECMASCRIPT = JAVASCRIPT***

♦ Exemple de integrare:

<script src="URL-fisier-Javascript.js"><script/>

<script>

/\*instrucțiuni Javascript\*/

<script/>

<marcaj eveniment="cod Javascript"><marcaj/>

Javascript - Tipuri de date

> Primitive:

String (lungime maximă 253-1): "text", 'alt text'

Number: -43, 2.32

Boolean: true, false

◇ Undefined: undefined

Null: null

Obiectuale:

Object: {}, {nume:"text", varsta:23}

Array: ["Ene", 23, 1.72]

> Date: new Date("2023-03-28")

> RegExp

✩ Speciale:

Function: function adunare(){ }

***VARIABILE ŞI OPERATORI***

→ Operatori:

var a1=-3, a2=5;

let b=2.3, p={x:2, y:3};

const constanta=3.14;

let b1=a1; // copierea valorii lui a1

b1-4; // nu afectează valoarea lui a1

let q=p; // copierea referinței lui p

q.x=1; // afectează valoarea lui p

> aritmetici: +, -, \*, /, %, ++, \*\*

◇ de atribuire: =, +=, \*\*=

◇ de comparare: ==,== !=, !==, <, <=, >=

logici: &&, ||,!

de biti: &, |, ~,^, <<, >>

de tip: typeof, instanceof

***COLECŢIILE SET ŞI MAP***

→ Colecţie cu elemente unice: Set

let s=new Set();

s.add(valoare);

s.size;

s.has (valoare);

s.delete(valoare);

s.values();

s.clear();

> Colecţie cu perechi cheie valoare: Map

let m=new Map();

m.set(cheie, valoare);

m.size;

m.has(cheie);

m.get(cheie);

m.values();

m.clear();

***FUNCŢII JAVASCRIPT***

> Definiţia scurtă a funcţiilor (arrow definition):

> nu are acces la colecția arguments pe care o întâlnim implicit la function, deoarece foloseşte exclusiv lista implementată a parametrilor;

> nu permite implementarea unui constructor;

this se referă la obiectul window

***OBIECTE JAVASCRIPT***

Proprietăţile publice se pot şterge:

delete p1.x;

→ Proprietăţile şi metodele private (ES2022) NU sunt se pot şterge!

Clasele se pot extinde mult mai uşor moştenindu-se proprietățile, metodele și constructorul (care se numeşte super)!

> În clase putem defini chiar şi metode statice pe care să le utilizăm la nivelul acesteia pentru că nu sunt specifice obiectelor.

Pentru a defini proprietăți private, care să nu poată fi modificate direct, acestea trebuie implementate în constructor împreună cu get şi set, după caz.

***PROPRIETĂȚI ȘI METODE PENTRU STRING***

→ Proprietăţi:

◇ length (lungimea unui string)

◇ prototype (definirea altor proprietăţi sau metode)

→ Metode:

replace (deCautat, inlocuire)

◇ slice(indexZ), substring(indexStart)

substring(indexStart, indexFinal)

split(separator)

Proprietăţi şi metode pentru Array

→ Proprietăţi:

◇ length (lungimea unui Array)

◇ prototype (definirea altor proprietăţi sau metode)

> Metode:

push(item), pop()

unshift(item), shift()

◊ splice (indexz, nrPozSterse, item)

◇ slice(indexZ, nrPozitii)

***METODELE SPLICE, UNSHIFT ŞI PUSH PENTRU ARRAY***

splice (indexz, nrPozSterse, item) metodă care inserează un item la poziția indexz şi şterge următoarele nrPozSterse poziții

push(item) metodă care inserează un item după ultimul element din array şi returnează numărul de elemente ale noului array format

unshift(item) metodă care inserează un item înaintea primului element din array şi returnează numărul de elemente ale noului array format

> Exemple:

arr.splice(0, 0, item) -> adaugă item înainte de prima poziție a array-ului arr

> arr.unshift (item) -> adaugă item pe prima poziţie a array-ului arr şi returnează numărul de elemente ale noului array

arr.splice(arr.length, 0, item) -> adaugă item după ultima poziție a array-ului arr

arr.push(item) -> adaugă item după ultima poziţie a array-ului arr şi returnează numărul de elemente ale noului array

***METODELE SPLICE, SHIFT ȘI POP PENTRU ARRAY***

✩ splice (indexZ, nrPozSterse) metodă care şterge nrPozSterse elemente începând cu poziţia indexz şi returnează un array cu acestea

→ pop() metodă care şterge ultimul element şi îl returnează

shift() metodă care şterge primul element şi îl returnează

> Exemple:

arr.splice(0, 1)-> şterge primul element al array-ului arr şi returnează un array cu elementul şters

arr.shift()-> şterge primul element şi îl returnează

arr.splice(-1, 1) -> şterge ultimul element al array-ului arr şi returnează un array cu elementul şters

arr.pop() -> şterge ultimul element și îl returnează

***OPERATORUL...SPREAD***

Definirea funcțiilor cu mai mulţi parametri:

function suma (...parametri) { /\* corpul funcției unde argumentul parametri va fi văzut de tip Array \*/ }

→ Concatenarea tablourilor de tip Array: let a=[1,2,3], b=[5,6];

let c=[...a,...b,7];

♦ Concatenarea proprietăților obiectelor: let a={x: 2}, b={y: 3}; let c= { ...a, ...b };

Se poate aplica împreună cu metodele splice, push, unshift pentru un Array: let a=[1,2,3], b=[5,6]; a.push(...b); //adaugă tabloul b la finalul tabloului a