Bazele HTML, CSS si JavaScript

HTML (HyperText Markup Language)

Este limbajul de bază folosit pentru a crea pagini web.

CSS (Cascading Style Sheets)

Este folosit pentru a stiliza paginile web create cu HTML

Concepte de bază HTML:

1. Structura de bază a unei pagini HTML:

```html
<!DOCTYPE html>
<html>
<head>
    <title>Titlul paginii</title>
</head>
<body>
    <!-- Conținutul paginii -->
</body>
</html>
```

2. Etichete HTML de bază:

<h1> până la <h6>: titluri de diferite niveluri

<p>: paragrafe de text

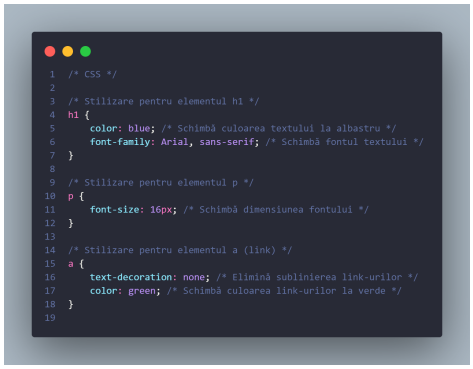
Textul link-ului: link-uri

<img src="calea-catre-imagine.jpg" alt="Text alternativ">: imagini

---

Concepte de bază CSS

1. Stilizarea elementelor HTML folosind CSS:



```css
/* CSS */

/* Stilizare pentru elementul h1 */
h1 {
    color: blue; /* Schimbă culoarea textului la albastru */
    font-family: Arial, sans-serif; /* Schimbă fontul textului */
}

/* Stilizare pentru elementul p */
p {
    font-size: 16px; /* Schimbă dimensiunea fontului */
}

/* Stilizare pentru elementul a (link) */
a {
    text-decoration: none; /* Elimină sublinierea link-urilor */
    color: green; /* Schimbă culoarea link-urilor la verde */
}
```

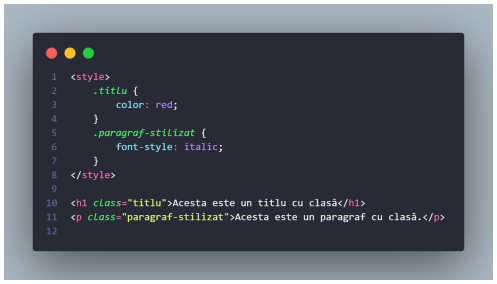
Crearea de tabele și clase în HTML:

1. Crearea unei tabele:

```html
<table border="1">
    <tr>
        <td>Coloana 1, Rând 1</td>
        <td>Coloana 2, Rând 1</td>
    </tr>
    <tr>
        <td>Coloana 1, Rând 2</td>
        <td>Coloana 2, Rând 2</td>
    </tr>
</table>
```

Aceasta creează o tabelă simplă cu 2 rânduri și 2 coloane.

2. Crearea claselor în HTML:



```html
<style>
    .titlu {
        color: red;
    }
    .paragraf-stilizat {
        font-style: italic;
    }
</style>

<h1 class="titlu">Acesta este un titlu cu clasă</h1>
<p class="paragraf-stilizat">Acesta este un paragraf cu clasă.</p>
```

În acest exemplu, am creat două clase CSS: .titlu și .paragraf-stilizat. Aceste clase pot fi apoi aplicate elementelor HTML pentru a le stiliza conform regulilor definite în CSS .

Aceasta este o introducere simplă în HTML, CSS și crearea de tabele și clase.

Cu aceste concepte de bază, puteți începe să construiți pagini web simple și să le stilizați conform preferințelor dvs.

CSS Flexbox și Grid Layout

CSS Flexbox

Flexbox este un model de design unidimensional care permite alinierea elementelor în orice direcție - orizontal sau vertical - și alinierea elementelor în raport cu containerul părinte.

Exemplu de utilizare a Flexbox pentru a alinia elemente într-un container orizontal:

```html
<style>
    .container {
        display: flex;
        justify-content: space-between; /* Aliniere la margini cu spațiu între elemente */
    }
    .element {
        flex: 1; /* Elementul ocupă spațiul disponibil în container */
        margin: 10px;
        padding: 20px;
        background-color: lightblue;
    }
</style>

<div class="container">
    <div class="element">Element 1</div>
    <div class="element">Element 2</div>
    <div class="element">Element 3</div>
</div>
```

CSS Grid Layout

CSS Grid Layout este un model de design bidimensional care permite definirea unui grid cu rânduri și coloane, oferind un control precis asupra layout-ului paginii.

Exemplu de utilizare a CSS Grid pentru a crea un layout cu rânduri și coloane

```html
<style>
    .container {
        display: grid;
        grid-template-columns: repeat(3, 1fr); /* 3 coloane egale */
        grid-gap: 10px; /* Spațiu între celule */
    }
    .element {
        padding: 20px;
        background-color: lightgreen;
        text-align: center;
    }
</style>

<div class="container">
    <div class="element">1</div>
    <div class="element">2</div>
    <div class="element">3</div>
    <div class="element">4</div>
    <div class="element">5</div>
    <div class="element">6</div>
</div>
```

Selecții mai complexe de clasă și ID:

Puteți utiliza selecții mai complexe pentru a specifica stilizarea pentru elemente specifice în funcție de clase sau ID-uri. De exemplu:

```
<style>
    .container p {
        font-size: 18px; /* Stilizare pentru paragrafe în interiorul elementului cu clasa "container" */
    }
    #element-special {
        background-color: yellow; /* Stilizare pentru elementul cu ID-ul "element-special" */
    }
</style>

<div class="container">
    <p>Aceasta este o paragrafă în interiorul unui container.</p>
    <p>Aceasta este o altă paragrafă în interiorul aceluiași container.</p>
</div>
<div id="element-special">
    Acest element are un fundal galben datorită ID-ului său unic.
</div>
```

Aceasta este o introducere în concepte mai avansate ale HTML și CSS. Cu aceste noțiuni, puteți începe să creați layout-uri complexe și să stilizați paginile web în mod flexibil și precis.

HTML Formular și CSS Stilizare

HTML Formular

```
<form>
    <label for="nume">Nume:</label>
    <input type="text" id="nume" name="nume"><br><br>
    <label for="parola">Parolă:</label>
    <input type="password" id="parola" name="parola"><br><br>
    <input type="submit" value="Trimite">
</form>
```

definitie

<label> este eticheta pentru a descrie ce se așteaptă de la utilizator în câmpul de input.

type="text" definește un câmp de text în care utilizatorul poate introduce text.

type="password" definește un câmp de text special pentru parole, în care caracterele sunt afișate ca buline sau asteriscuri.

CSS Stilizare pentru Formular cu Comentarii

```
form {
    width: 300px; /* Lățimea formularului */
    margin: 0 auto; /* Alinierea formularului în centru pe pagină */
    padding: 20px; /* Spațiul în interiorul formularului */
    border: 1px solid #ccc; /* Bordura gri de 1 pixel */
    border-radius: 10px; /* Colțuri rotunjite de 10 pixeli */
}

label {
    display: block; /* Așează etichetele pe linii separate */
    margin-bottom: 10px; /* Spațiu între etichete */
}

input[type="text"],
input[type="password"] {
    width: 100%; /* Lățimea completă a câmpului de input */
    padding: 10px; /* Spațiul în interiorul câmpului de input */
    margin-bottom: 15px; /* Spațiu între câmpuri */
    border: 1px solid #ccc; /* Bordura gri de 1 pixel */
    border-radius: 5px; /* Colțuri rotunjite de 5 pixeli */
}

input[type="submit"] {
    background-color: #4caf50; /* Culoarea de fundal a butonului de submit */
    color: white; /* Culoarea textului din buton */
    border: none; /* Eliminarea bordurii */
    padding: 15px 20px; /* Spațiul în interiorul butonului */
    border-radius: 5px; /* Colțuri rotunjite de 5 pixeli */
    cursor: pointer; /* Indicator de cursor pentru buton */
}
```

Aceasta este o formă HTML simplă stilizată cu CSS. Puteți adăuga mai multe câmpuri și personaliza stilurile după preferințe.

CSS Animatii

Exemplu de animație în CSS

```css
@keyframes schimbare-culoare {
    0% {
        background-color: red; /* La 0%, culoarea de fundal este roșie */
    }
    50% {
        background-color: yellow; /* La 50%, culoarea de fundal este galbenă */
    }
    100% {
        background-color: green; /* La 100%, culoarea de fundal este verde */
    }
}

/* Stilizarea elementului <div> */
div {
    width: 100px; /* Lățimea elementului este de 100 de pixeli */
    height: 100px; /* Înălțimea elementului este de 100 de pixeli */
    background-color: red; /* Culoarea de fundal inițială a elementului este roșie */
    animation: schimbare-culoare 3s infinite; /* Animație de 3 secunde, repetată la infinit */
    /* Animation: numele animației, durata, modul de repetare */
}

```

Definirea animației folosind @keyframes

Media Queries

CSS cu Media Queries pentru Design Receptiv

```css
/* Media query pentru ecrane cu lățimea maximă de 600 de pixeli */
@media only screen and (max-width: 600px) {
    /* Regula CSS: reducerea dimensiunii fontului pentru elementul <body> */
    body {
        font-size: 14px; /* Dimensiunea fontului este setată la 14 pixeli */
    }
    /* Puteți adăuga și alte stilizări pentru ecrane mici aici */
}

```

JavaScript și Interactivitatea Paginilor Web

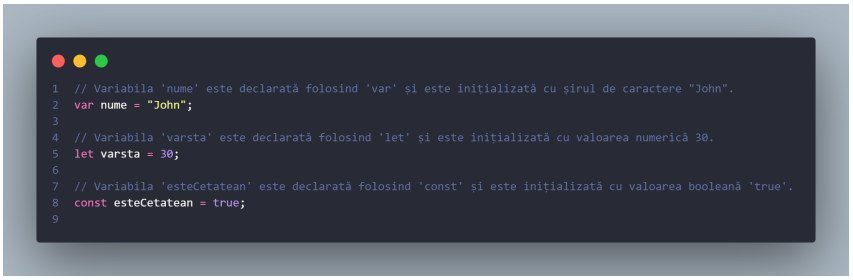
JavaScript este un limbaj de programare de bază pentru dezvoltarea web și este utilizat pentru a adăuga interactivitate paginilor web.
De exemplu, aici este un exemplu de cum puteți folosi JavaScript pentru a schimba conținutul unei pagini în funcție de acțiunile utilizatorului

```html
<!DOCTYPE html>
<html>
<head>
    <title>Interactivitate JavaScript</title>
</head>
<body>

<button onclick="schimbaTextul()">Apasă aici</button>

<p id="schimba">Aceasta este o propoziție.</p>

<script>
function schimbaTextul() {
    document.getElementById("schimba").innerHTML = "Noul text schimbat cu JavaScript!";
}
</script>

</body>
</html>

```

În acest exemplu, JavaScript este folosit pentru a schimba textul unui paragraf atunci când utilizatorul apasă pe buton.

Variabile

Variabilele sunt utilizate pentru a stoca date.
Puteți declara o variabilă în JavaScript folosind cuvântul cheie var, let sau const

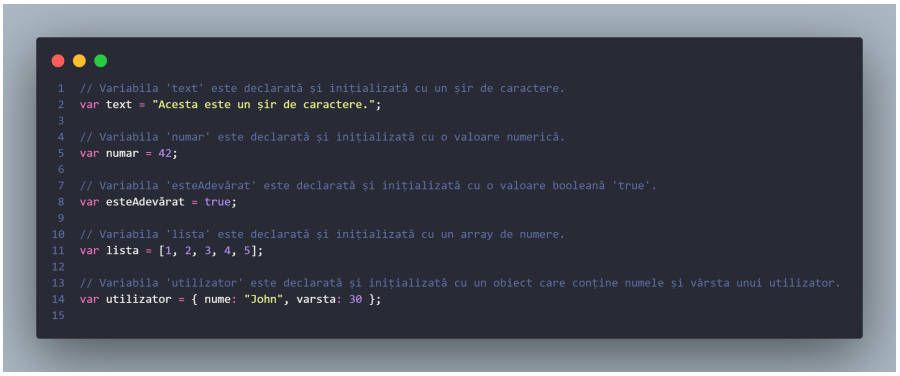


```
// Variabila 'nume' este declarată folosind 'var' și este inițializată cu șirul de caractere "John".
var nume = "John";

// Variabila 'varsta' este declarată folosind 'let' și este inițializată cu valoarea numerică 30.
let varsta = 30;

// Variabila 'esteCetatean' este declarată folosind 'const' și este inițializată cu valoarea booleană 'true'.
const esteCetatean = true;

```

Tipuri de Date

JavaScript are diferite tipuri de date, cum ar fi șiruri de caractere

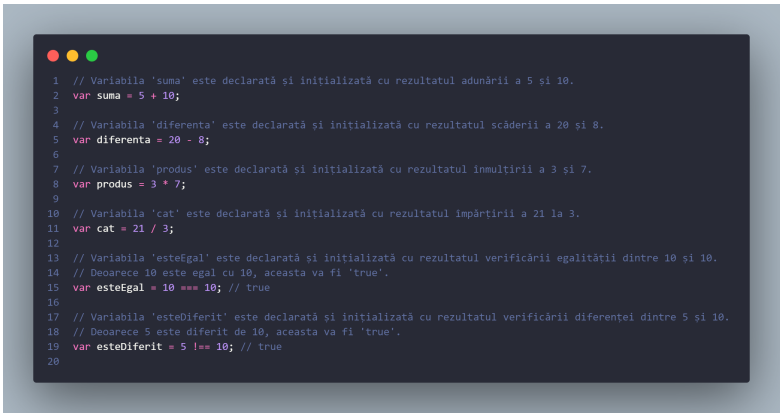
(strings), numere (numbers), booleeni (booleans), tablouri (arrays), obiecte (objects), etc.



```
// Variabila 'text' este declarată și inițializată cu un șir de caractere.
var text = "Acesta este un șir de caractere.";

// Variabila 'numar' este declarată și inițializată cu o valoare numerică.
var numar = 42;

// Variabila 'esteAdevărat' este declarată și inițializată cu o valoare booleană 'true'.
var esteAdevărat = true;

// Variabila 'lista' este declarată și inițializată cu un array de numere.
var lista = [1, 2, 3, 4, 5];

// Variabila 'utilizator' este declarată și inițializată cu un obiect care conține numele și vârsta unui utilizator.
var utilizator = { nume: "John", varsta: 30 };

```

Operatori

JavaScript folosește operatori pentru a efectua operații pe variabile și valori



```
// Variabila 'suma' este declarată și inițializată cu rezultatul adunării a 5 și 10.
var suma = 5 + 10;

// Variabila 'diferenta' este declarată și inițializată cu rezultatul scăderii a 20 și 8.
var diferenta = 20 - 8;

// Variabila 'produs' este declarată și inițializată cu rezultatul înmulțirii a 3 și 7.
var produs = 3 * 7;

// Variabila 'cat' este declarată și inițializată cu rezultatul împărțirii a 21 la 3.
var cat = 21 / 3;

// Variabila 'esteEgal' este declarată și inițializată cu rezultatul verificării egalității dintre 10 și 10.
// Deoarece 10 este egal cu 10, aceasta va fi 'true'.
var esteEgal = 10 === 10; // true

// Variabila 'esteDiferit' este declarată și inițializată cu rezultatul verificării diferenței dintre 5 și 10.
// Deoarece 5 este diferit de 10, aceasta va fi 'true'.
var esteDiferit = 5 !== 10; // true

```

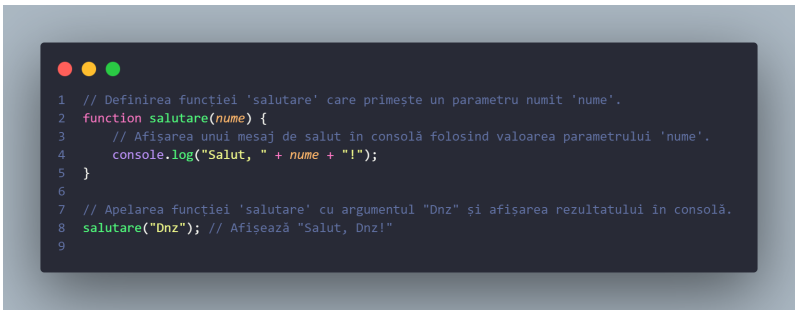
Structuri de Control

JavaScript oferă structuri de control, cum ar fi if, else, for, while și switch pentru a controla fluxul programului.

```javascript
// Declararea și inițializarea variabilei 'numar' cu valoarea 15.
var numar = 15;

// Verificarea dacă 'numar' este mai mare decât 10 și afișarea unui mesaj corespunzător în consolă.
if (numar > 10) {
    console.log("Numărul este mai mare decât 10.");
} else {
    console.log("Numărul este mai mic sau egal cu 10.");
}

// Utilizarea buclei 'for' pentru a afișa un mesaj de iterație de la 0 la 4.
for (var i = 0; i < 5; i++) {
    console.log("Iterație numărul " + i);
}

// Utilizarea buclei 'while' pentru a afișa 'numar' și a decrementa 'numar' până când devine 0.
while (numar > 0) {
    console.log(numar);
    numar--;
}

```

Funcții

Funcțiile sunt utilizate pentru a defini blocuri de cod reutilizabile.



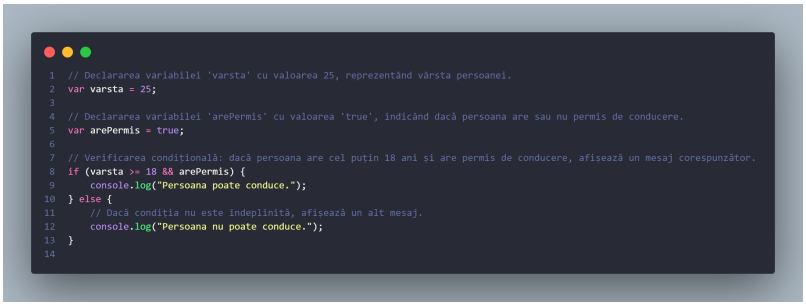
```javascript
// Definirea funcției 'salutare' care primește un parametru numit 'nume'.
function salutare(nume) {
    // Afișarea unui mesaj de salut în consolă folosind valoarea parametrului 'nume'.
    console.log("Salut, " + nume + "!");
}

// Apelarea funcției 'salutare' cu argumentul "Dnz" și afișarea rezultatului în consolă.
salutare("Dnz"); // Afișează "Salut, Dnz!"

```

Operatori Logici

Operatorii logici sunt esențiali în JavaScript și sunt utilizați pentru a efectua operații logice între două sau mai multe expresii.

Operatori Logici de Și (`&&`)

Operatorul && este utilizat pentru a verifica dacă ambele expresii sunt adevărate (true).
Dacă ambele expresii sunt adevărate, atunci întreaga expresie este adevărată.

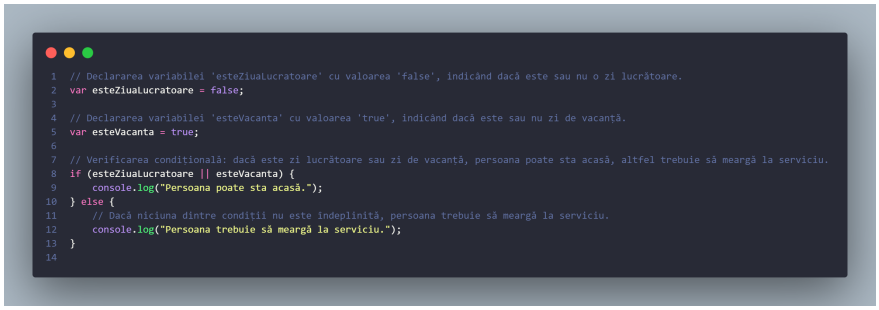


```javascript
// Declararea variabilei 'varsta' cu valoarea 25, reprezentând vârsta persoanei.
var varsta = 25;

// Declararea variabilei 'arePermis' cu valoarea 'true', indicând dacă persoana are sau nu permis de conducere.
var arePermis = true;

// Verificarea condițională: dacă persoana are cel puțin 18 ani și are permis de conducere, afișează un mesaj corespunzător.
if (varsta >= 18 && arePermis) {
    console.log("Persoana poate conduce.");
} else {
    // Dacă condiția nu este îndeplinită, afișează un alt mesaj.
    console.log("Persoana nu poate conduce.");
}

```

Această structură `if-else` verifică dacă `varsta` este mai mare sau egală cu 18 și dacă `arePermis` este true.
Dacă ambele condiții sunt îndeplinite, mesajul "Persoana poate conduce." va fi afișat în consolă.
Altfel, mesajul "Persoana nu poate conduce."

Operator Logica de Sau (`||`)

Operatorul || este utilizat pentru a verifica dacă cel puțin una dintre expresii este adevărată.
Dacă cel puțin o expresie este adevărată, întreaga expresie este adevărată.



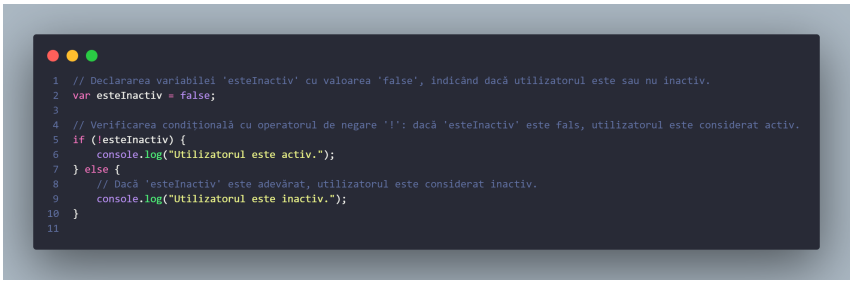
```
// Declararea variabilei 'esteZiuaLucratoare' cu valoarea 'false', indicând dacă este sau nu o zi lucrătoare.
var esteZiuaLucratoare = false;

// Declararea variabilei 'esteVacanta' cu valoarea 'true', indicând dacă este sau nu zi de vacanță.
var esteVacanta = true;

// Verificarea conditională: dacă este zi lucrătoare sau zi de vacanță, persoana poate sta acasă, altfel trebuie să meargă la serviciu.
if (esteZiuaLucratoare || esteVacanta) {
    console.log("Persoana poate sta acasă.");
} else {
    // Dacă niciuna dintre condiții nu este îndeplinită, persoana trebuie să meargă la serviciu.
    console.log("Persoana trebuie să meargă la serviciu.");
}
```

Type your text

În acest exemplu, `||` reprezintă operatorul logic "sau".
Dacă cel puțin una dintre variabile (`esteZiuaLucratoare` sau `esteVacanta`) este `true`, mesajul "Persoana poate sta acasă." va fi afișat.
Altfel, mesajul "Persoana trebuie să meargă la serviciu." va fi afișat.

Operator Logica de Negare (`!`)

Operatorul ! este utilizat pentru a nega o expresie. Dacă o expresie este adevărată, ! o face falsă, și invers.



```
// Declararea variabilei 'esteInactiv' cu valoarea 'false', indicând dacă utilizatorul este sau nu inactiv.
var esteInactiv = false;

// Verificarea conditională cu operatorul de negare '!': dacă 'esteInactiv' este fals, utilizatorul este considerat activ.
if (!esteInactiv) {
    console.log("Utilizatorul este activ.");
} else {
    // Dacă 'esteInactiv' este adevărat, utilizatorul este considerat inactiv.
    console.log("Utilizatorul este inactiv.");
}
```

În acest exemplu, `!` reprezintă operatorul de negare logică.
Dacă variabila `esteInactiv` este `false` (adică negarea sa este `true`), atunci mesajul "Utilizatorul este activ." va fi afișat.
Altfel, dacă `esteInactiv` este `true` (adică negarea sa este `false`), mesajul "Utilizatorul este inactiv." va fi afișat.

Dragi cursanți,

Faptul că ați explorat tainele dezvoltării web prin HTML, CSS și JavaScript este cu adevărat remarcabil.

Vă mulțumesc pentru entuziasmul și dedicarea voastră în această călătorie captivantă a cunoașterii.

Ați creat o bază solidă pentru viitorul vostru în lumea dezvoltării web.

Felicitări pentru realizări și vă încurajez să continuați să explorați și să inovați în acest domeniu fascinant.

Cu recunoștință,

[DNZ]
[Autorul/Creatorul cursului "Bazele HTML, CSS și JavaScript"]

dnz