1)

1. **<html>**: Səhifənin əsas elementidir və bütün digər elementləri əhatə edir.
2. **<head>**: Səhifənin başlığını və meta məlumatlarını (səhifənin dilini, stil məlumatlarını və s.) daxil edir.
3. **<title>**: Səhifənin başlığını təyin edir, ki bu başlıq brauzer pəncərəsində görünəcək.
4. **<body>**: Səhifənin asılı hissəsini, yəni məzmunu təyin edir. Mətn, şəkillər, linklər və daha çoxunu bu hissədə əlavə edirsiniz.
5. **<h1>**, **<h2>**, ... **<h6>**: Başlıq elementləridir və səhifədə başlıqları hiearşik olaraq təyin etmək üçün istifadə olunur. **<h1>** ən böyük başlıqdır, **<h6>** isə ən kiçikdir.
6. **<p>**: Mətn paragraflarını təyin etmək üçün istifadə olunur.
7. **<a>**: Hiperlinklər (linklər) təyin etmək üçün istifadə olunur.
8. **<img>**: Şəkil elementidir və şəkillərin və ya digər multimedia elementlərin səhifəyə əlavə edilməsinə kömək edir.
9. **<ul>** və **<ol>**: Siyahılar yaratmaq üçün istifadə olunurlar. **<ul>** tərs sıralanmış siyahını (məs. mətn siyahısı), **<ol>** isə nömrələnmiş siyahını (məs. rəqəmli siyahı) təyin edir.
10. **<div>**: Ümumi məzmun bölmələrinin təyin edilməsi üçün istifadə olunur. CSS ilə stil vermək və JavaScript ilə davranışları idarə etmək üçün əlavə struktur yaratmaq üçün istifadə edilir.

2)

**Block-Level Elementlər:**

1. **Ümumi Görünüş:** Block-level elementlər öz başına bir xətt tuturlar və səhifədə yeni bir blok (blokqutu) təşkil edir. Bu elementlər genişlikləri səhifənin tam eninə çıxır və yeni bir sətirə keçirlər.
2. **Mətn**: Block elementləri ümumiyyətlə mətn və digər block və inline elementlərini özündə əhatə edə bilər. Mətn elementləri mətni yeni sətirdə başladar və bitirir.
3. **Padding və Margin:** Block-level elementlər üçün padding və margin təyin edilə bilər, bu da onların arasında boşluq yaradır.
4. **Örnek Elementlər:** **<div>**, **<p>**, **<h1>**, **<ul>**, **<li>**, **<table>**, **<form>** kimi elementlər block-level elementlərə məxsus nümunələrdir.

**Inline Elementlər:**

1. **Ümumi Görünüş:** Inline elementlər, blokların içində yerləşən və bir sətirin içində qalmış elementlərdir. Onlar yalnız təzyiq olunan mətni təyin etmək üçün istifadə olunurlar.
2. **Mətn:** Inline elementlər sadəcə mətn və inline elementləri əhatə edirlər. Mətni aşağı-qırmızı, bold və ya link kimi formatlaya bilərlər.
3. **Padding və Margin:** Inline elementlər üçün padding və margin təyin edilə bilməz, və onların arasında boşluq yaranmaz.
4. **Örnek Elementlər:** **<span>**, **<a>**, **<strong>**, **<em>**, **<img>**, **<br>** kimi elementlər inline elementlərə məxsus nümunələrdir.

Mühüm olan şey, block və inline elementlərin uyğun yerləşdirilməsi və məzmunun strukturunu düzgün təşkil etməkdir. Blok elementlər əsasən maddənin bölmələrinin təyin edilməsində və düzgün strukturun qurulmasında istifadə edilir, əmələ inline elementlər isə mətni və onun görünüşünü təyin etməkdə istifadə edilir.

Начало формы

3)

Birnci inline) <p style="color: blue; font-size: 16px;">Mavi mətn</p>

İkinci internal) <html>

<head>

<style>

p {

color: blue;

font-size: 16px;

}

</style>

</head>

<body>

<p>Mavi mətn</p>

</body>

</html>

Uuchncu exteernal) <html>

<head>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

<p>Mavi mətn</p>

</body>

</html>

Js birnci inline) <p onclick="alert('Salam, dünya!')">Salam, dünya!</p>

İkinci internal) <html>

<head>

<script>

function salamDunya() {

alert('Salam, dünya!');

}

</script>

</head>

<body>

<p onclick="salamDunya()">Salam, dünya!</p>

</body>

</html>

Uchuncu external) <html>

<head>

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

</head>

<body>

<p onclick="salamDunya()">Salam, dünya!</p>

</body>

</html>

4)

<button id="myButton" onclick="doSomething()">Əməliyyatı yerinə yetir</button>

Box modeli-**Content,** **Padding,** **Border,** **Margin,** **Padding**

**5) Block (display: block) (div,p,ol,ul,li)**

* Blok elementlər, öz başına bir xətt tuturlar və səhifədə yeni bir blok (blokqutu) təşkil edirlər.
* Yatayda genişləyərək sıradan sonra gələn elementlər yeni sətrə yerləşir.
* Blok elementlərin özəllikləri, genişlikləri varsayılan olaraq səhifənin tam genişliyinə çıxar.

**Inline (display: inline): (span,a)**

* İnline elementlər sıradan bir-birinə birləşmiş və yan-yana sıralanmış kimi görünür.
* Dikeydə yer tutmurlar, yəni bir növbədəki element sıradan sonra davam edir.
* İnline elementlərin genişliyi və hündürlüyü içəriyə uyğun gələn məzmunun ölçüsünə görə avtomatik olaraq ayarlanır.
* **Inline-Block (display: inline-block)(button)**
  + İnline-blok, inline və block elementlərin bəzilərini bir araya gətirən bir növ kombinasiyadır.
  + İnline-blok elementlər sıradan bir-birinə birləşmiş kimi görünür, lakin genişlik və hündürlük kimi blok elementlərin xüsusiyyətlərini qəbul edə bilər.
  + İnline-blok, eyni sıradakı digər elementlərlə birgə sıraya yerləşir, lakin özünün içindəki məzmunun ölçüsünü özü təyin edir.

6)

**Flex Grow (flex-grow):**

* **flex-grow** xüsusiyyəti, bir elementin boşluğu doldurmaq üçün ne qədər böyük ola biləcəyini təyin edir.
* Əgər bir containerin içindəki elementlər arasında boşluq qalırsa, **flex-grow** dəyəri böyüklüyünü əsasında, hər elementin nisbətində paylanır.
* Default olaraq, bütün elementlərin **flex-grow** dəyəri 0-dir, yəni boşluq olarsa bu elementlər genişləməz.

.container {

display: flex;

}

.item {

flex-grow: 1;

}

**Flex Shrink (flex-shrink):**

* **flex-shrink** xüsusiyyəti, bir elementin sıxışmaq istədiyi halda ne qədər küçülməli olduğunu təyin edir.
* Əgər containerin içindəki elementlər, mövcud boşluğu doldurmağa kifayət qədər kiçikdir, **flex-shrink** dəyərinə əsasən, hər element özünü sıxışdırmağa başlayır.
* Default olaraq, bütün elementlərin **flex-shrink** dəyəri 1-dir, yəni elementlər sıxışmaq üçün eyni nisbətdə küçülər.

.container {

display: flex;

}

.item {

flex-shrink: 1;

}

**Flex Basis (flex-basis):**

* **flex-basis** xüsusiyyəti, bir elementin əsas (initial) ölçüsünü təyin edir.
* **flex-basis**, əsas olaraq, bir elementin minimum ölçüsünü müəyyən edir, lakin bu ölçü, containerin içində digər elementlərin əsas ölçüləri və sıxışma və ya genişləmə dəyərləri ilə dəyişdirilə bilər.

.container {

display: flex;

}

.item {

flex-basis: 100px;

}

7)

**Grid Container (Qutu):** Bir grid yaratmaq üçün əsas element grid container (grid qutusu) olmalıdır. Grid container, bütün grid elementlərini əhatə edən bir div və ya başqa bir HTML elementi olur. Grid containerə **display: grid;** tətbiq edilir.

**Grid Items (Elementlər):** Gridin içində olan elementlər grid items (grid elementləri) kimi adlandırılır. Bu elementlər grid containerin içində yerləşir və gridin hər hansı bir hissəsində yerləşdirilə bilər.

**Grid Columns və Rows (Sütunlar və Sıralar):** CSS Grid Layout, sütunlar və sıralar təyin etmək üçün istifadə olunan grid-template-columns və grid-template-rows xüsusiyyətlərini təklif edir. Bu, sütun və sıra sayısını, hündürlüklərini və digər cür ölçülərini təyin etməyə imkan verir.

.grid-container {

display: grid;

grid-template-columns: 1fr 1fr 1fr; /\* Üç sütun \*/

grid-template-rows: 100px 200px; /\* İki sıra \*/

}

**Grid Gap (Boşluq):** Grid elementlərinin arasına boşluq əlavə etmək üçün grid-gap xüsusiyyəti istifadə olunur. Bu, sıraların və sütunların arasında məsafə təyin etmək üçün istifadə olunur

**Grid Template Areas (Şablon Əraziləri):** Grid elementlərini bir neçə bölməyə bölmək üçün grid-template-areas xüsusiyyəti istifadə olunur. Bu, sıraları və sütunları adlandırmaq və onlara veb səhifələrin quruluşunu təyin etmək üçün çox möhkəm bir alətdir.

.grid-container {

grid-template-areas:

"header header header"

"sidebar content content"

"footer footer footer";

}

8)

<div class="container">

<div class="block">Məzmun</div>

</div>

I variant grid)

.container {

display: grid;

width: 300px;

height: 300px;

place-items: center;

}

.block {

width: 150px;

height: 150px;

background-color: #3498db;

color: #fff;

text-align: center;

line-height: 150px;

}

II variant display block) .container {

width: 300px;

height: 300px;

display: flex;

justify-content: center;

align-items: center;

}

.block {

width: 150px;

height: 150px;

background-color: #3498db;

color: #fff;

text-align: center;

line-height: 150px;

}

III variant)

.container {

width: 300px;

height: 300px;

position: relative;

}

.block {

width: 150px;

height: 150px;

background-color: #3498db;

color: #fff;

text-align: center;

line-height: 150px;

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

}

9)

<div class="container">

<div class="item">1</div>

<div class="item">2</div>

<div class="item">3</div>

</div>

.container {

display: grid;

grid-template-columns: 1fr 2fr 1fr; /\* Sırasıyla, bir fr, iki fr, və bir fr genişliyə sahib sütunlar \*/

width: 1100px; /\* Containerın eni \*/

margin: 0 auto; /\* Containerı mərkəzləşdir \*/

gap: 10px; /\* Sıraların və sütunların arasında boşluq \*/

}

.item {

background-color: #3498db;

color: #fff;

text-align: center;

line-height: 150px;

}

10)

#content {

    animation: fadeIn 1s ease-in-out forwards; /\* fadeIn adlı animasiya tətbiq edin \*/

  }

  @keyframes fadeIn {

    from {

      opacity: 0;

    display: none; /\* İlk olaraq məzmunu gizləyin \*/

    }

    to {

      display: block; /\* Animasiyanın sonunda məzmunu görünür edin \*/

      opacity: 1;

    }

  }

11)

**String (Mətn):** String tipləri mətni ifadə etmək üçün istifadə olunur. Mətnlər tirav və ya dubl tirav ('' və ya "") içərisində yazılır.

1. **Number (Ədəd):** Number tipləri riyazi ədədləri ifadə etmək üçün istifadə olunur. Bu ədədlər tam ədədlər və onluq ədədlər (köhnəlmiş desimal sistemi) ola bilər.

**Boolean (Məntiqi):** Boolean tipləri yalnız iki dəyəri təmsil edir: **true** və **false**. Məntiqi tiplər şərtlər və nöqtələşmə əməliyyatlarında çox istifadə olunur.

1. **Array (Sıra):** Array tipləri məlumatları bir sıra (array) şəklində saxlamağa imkan verir. Arraylər elementlərdən ibarət sıralardır və hər bir elementi bir indeks ilə identifikasiya edirlər. **Object (Obyekt):** Object tipləri verilənləri açar-qiymət çiftləri ilə təmsil edir. Hər bir açar bir xüsusi dəyərə əsasən obyektin daxilindəki məlumatı əldə etmək və dəyişdirmək üçün istifadə olunur.
2. **Undefined və Null:** JavaScript-in öz xüsusi tipləri "undefined" və "null" dır. "undefined" dəyəri dəyişkənin təyin olunmayıb istifadə edildiyi halı nümayiş etdirir, və "null" dəyəri vacib bir dəyişənin dəyərinin mənasız olduğunu təyin etmək üçün istifadə olunur.

* 12) "Number" tipləri riyazi ədədləri təmsil edir və riyazi əməliyyatlar üçün nəzərdə tutulur.
* "String" tipləri mətn və simvolları təmsil edir və mətn əməliyyatları üçün nəzərdə tutulur.

Number🡪string

var ədəd = 42;

var mətn = String(ədəd); // Ədədi mətnə çevirir

console.log(mətn); // "42"

veya

var ədəd = 42;

var mətn = ədəd.toString(); // Ədədi mətnə çevirir

console.log(mətn); // "42"

string🡪number

var mətn = "42";

var ədəd = Number(mətn); // Mətni ədədə çevirir

console.log(ədəd); // 42

veya

var mətn = "42";

var ədəd = parseInt(mətn); // Mətni tam ədədə çevirir

console.log(ədəd); // 42

13) var tarix = new Date();

var gün = tarix.getDate();

console.log(gün); // Hazırda olan tarixdəki günlər (1-31)

14)znayu

15)

<button id="btn" data-color="red">Dəyişmək</button>

// HTML elementini JavaScript-də seçirik

var button = document.getElementById("btn");

// Atribut oxumaq üçün getAttribute

var color = button.getAttribute("data-color");

console.log("Atribut oxumaq: " + color); // "red"

// Atributı təyin etmək üçün setAttribute

button.setAttribute("data-color", "blue");

console.log("Atributı təyin etmək: " + button.getAttribute("data-color")); // "blue"

16) **push(),pop(),shift(),unshift(),concat()**

**17) for (var i = 1; i <= 10; i++) {**

**console.log(i);**

**}**

///////////

var i = 1;

while (i <= 10) {

console.log(i);

i++;

}

///////////

var i = 1;

do {

console.log(i);

i++;

} while (i <= 10);

18)

const sayılar = [1, 2, 3, 4, 5];

const kareleri = sayılar.map(sayı => sayı \* sayı);

console.log(kareleri); // [1, 4, 9, 16, 25]

//////////////////////////////

const sayılar = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

const tekSayılar = sayılar.filter(sayı => sayı % 2 === 1);

console.log(tekSayılar); // [1, 3, 5, 7, 9]

////////////////////////////////////////////

const sayılar = [1, 2, 3, 4, 5];

const toplam = sayılar.reduce((accumulator, sayı) => accumulator + sayı, 0);

console.log(toplam); // 15

19) var metin = "hello world";

var ilkHerfBöyük = metin.charAt(0).toUpperCase() + metin.slice(1);

console.log(ilkHerfBöyük); // "Hello world"

20) function pozitifOrta(dizi) {

var pozitifler = dizi.filter(function (element) {

return element > 0; // Pozitif tam sayıları filtrele

});

if (pozitifler.length === 0) {

return 0; // Pozitif tam sayı yoksa ortalama sıfırdır.

}

var toplam = pozitifler.reduce(function (toplam, sayı) {

return toplam + sayı; // Pozitif tam sayıların toplamını hesapla

}, 0);

var ortalama = toplam / pozitifler.length;

return ortalama;

}

var dizi = [3, -2, 8, -4, 5, -1, 9];

var ortalama = pozitifOrta(dizi);

console.log("Pozitif ədədlərin ortalaması: " + ortalama);

* 21) For Mobile devices: 320px-480px
* For Tablets or iPad: 480px - 768px
* For Laptop or small-size screen: 768px -1024px
* For Desktop or large-size screen: 1024px -1200px
* For Extra-large size device: 1200px and more