**BLOCK LEVEL ve INLINE LEVEL ELEMENTS**

Block level elementler %100 ünü kaplar satırın fakat inline elementler sadece gözüktükleri veya içindekiler kadar alanları kaplarlar. Örnek: DİV block elementtir. SPAN inline elementtir.

<https://www.w3schools.com/html/html_blocks.asp>

**ENTITIES**

Html kodlarımızın ıcınde bosluk bırakmak ıstersek bırden fazla bunu html algılamaz. Fakat entities ler bunun gibi durumlarda bize yardımcı olurlar. Ornegın &nbsp; yaparak bir tane space, defalarca yazarak istediğimiz kadar space bırakabiliriz.

Ayrıca Copyright sembollerini, döviz sembollerini, iskambil kağıtı işaretlerini, ve daha bir çok özel sembolü entities lerle yapabiliriz.

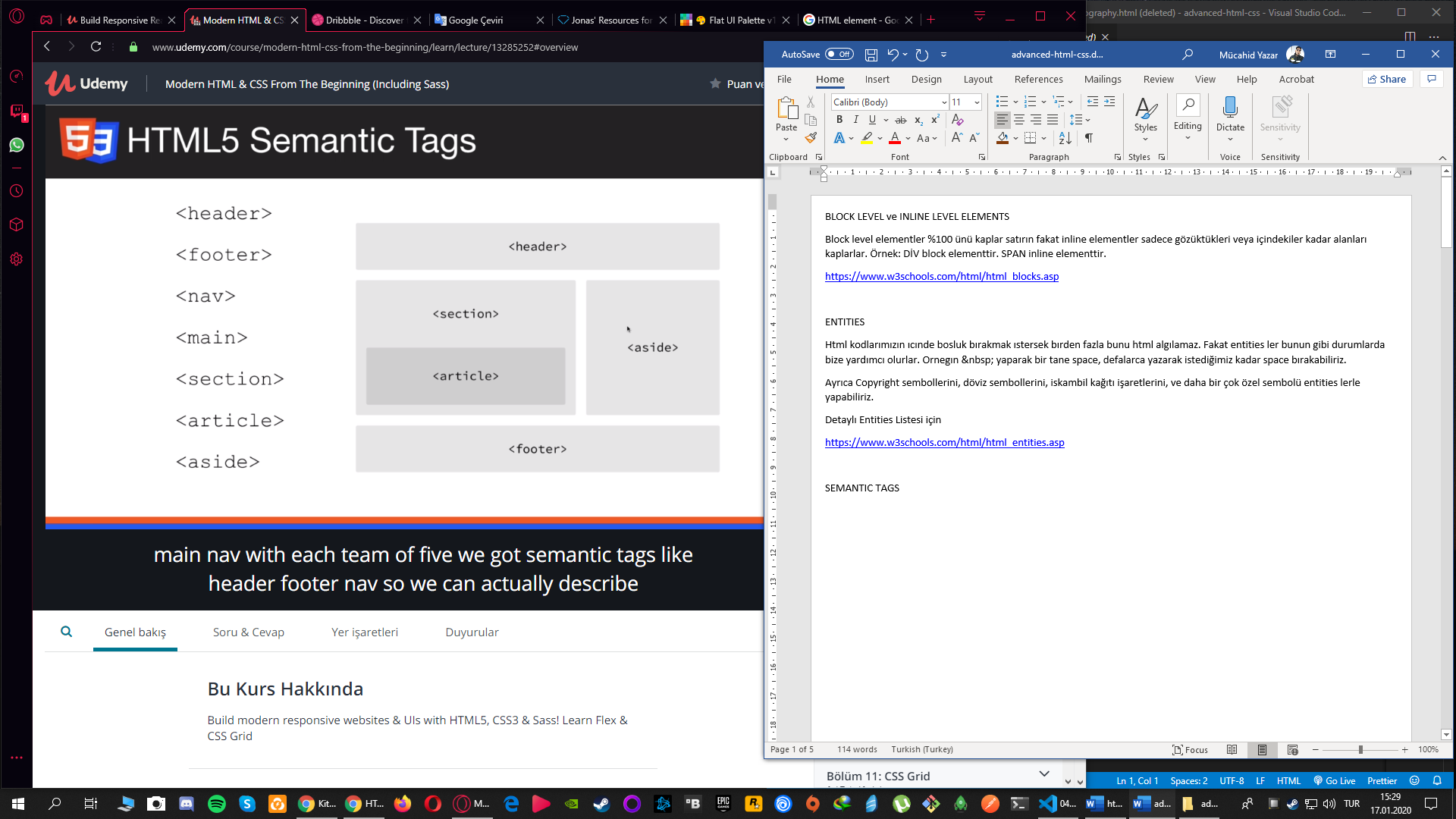
Detaylı Entities Listesi için

<https://www.w3schools.com/html/html_entities.asp>

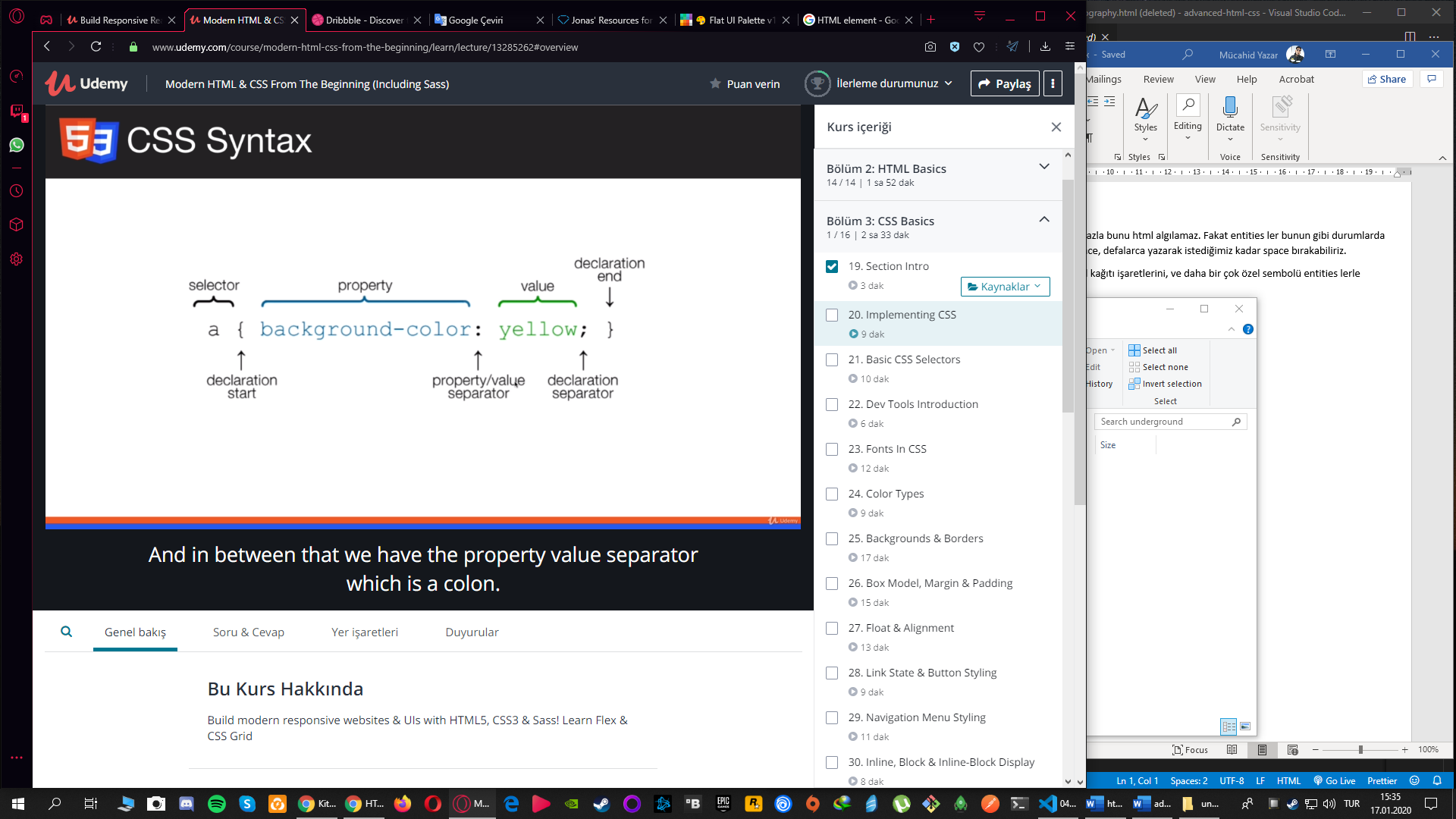
**SEMANTIC TAGS**

Sayfalarımızdaki düzene daha iyi bir anlam kazandırır.

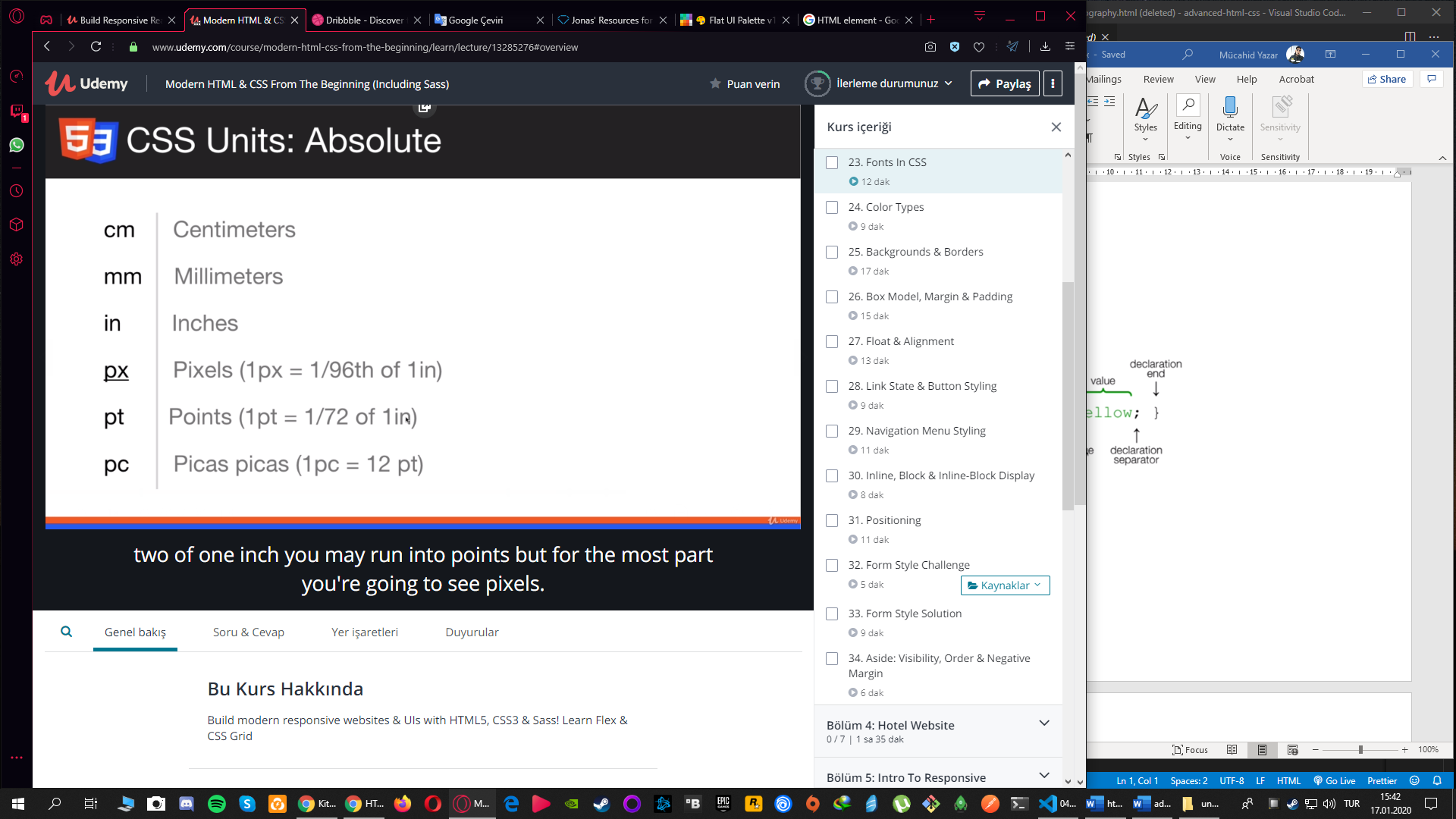
En popüler semantic tagler = header, footer, nav, main, section, article, aside

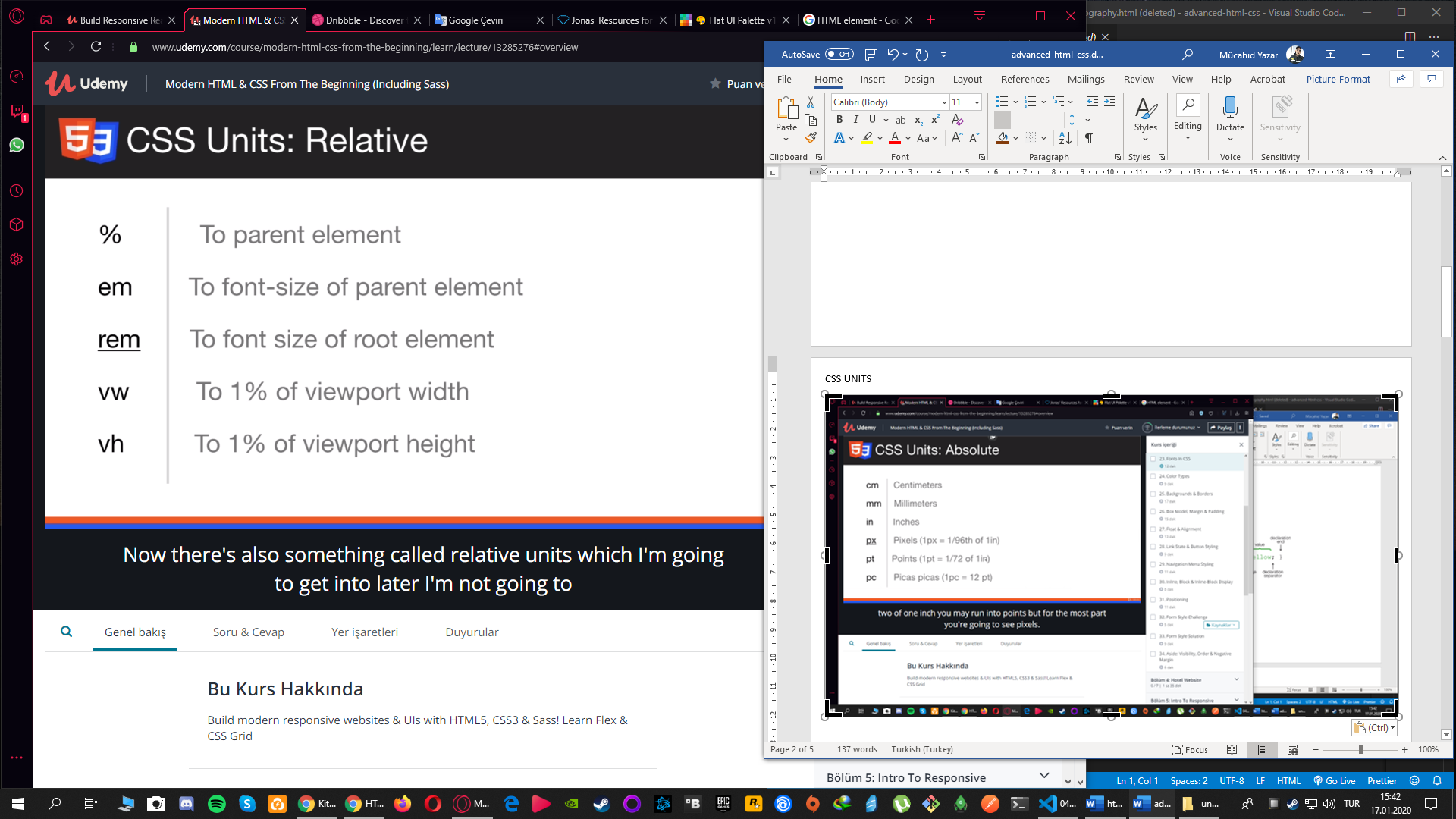


**CSS Syntax**

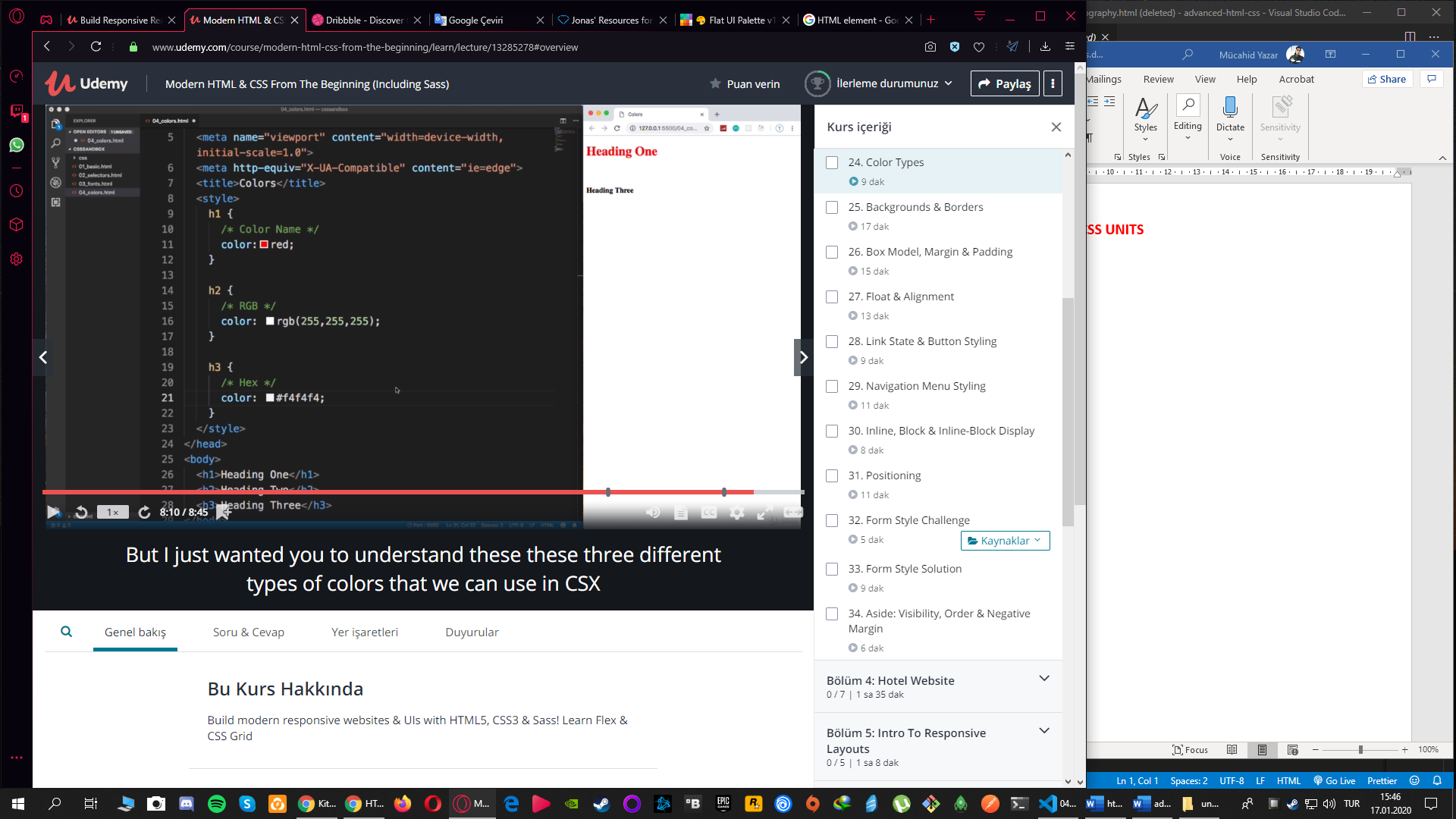


**CSS UNITS**





COLOR TYPES



**BACKGROUND**

Asagıdakı resımde yorumdaki satırda yapılan işlemlerin hepsi aşağıda tek satırda background property si altında yapılmıştır.

İlk center horizontal yani yatar, ikinci center vertical yani dikey olarak ortalama yapar.

Norepeat resmin tekrar etmemesini sağlar.

Cover Resmin alanı kaplamasını sağlar.

    #box-3 {

      /\* background-image: url(‘./img/stars.jpg’);

      background-repeat: no-repeat;

      background-position: center top;

      background-size: cover; \*/

      background: url(‘./img/stars.jpg’) no-repeat center center/cover;

      color: #fff;

      height: 300px;

      width: 600px;

    }

Background-attachment: fixed ise resmin ayarlandıgı yerde sürekli sabit kalmasını ne kadar sayfada aşağı insek de aynı yerde kalmasını sağlar.

    #box-4 {

      background: url('./img/leaf.png') no-repeat center center;

      background-attachment: fixed;

    }

**POSITION**

z-index: Öncelik sıralamasını belirtir. Z-index kimin yüksekse aynı pozisyondaki cisimlerde o en üstte gözükür.

Position: static = Default olarak bu deger kullanılır css tarafından.

Position: relative = Ana pozisyondur. Genelde Parent elemanlara göre başka elemanları pozisyon aldırılırken kullanılır. Eğer position relative iken top:50px dersek yukarıdan 50px aşağı düşer. Left:50 px dersek baktıgımız pencerenin sol bitiş hizasından 50px sağa kayar.

Position: absolute = Ana maın yani parent elemant relative olarak belirlendikten sonra, o elemana göre pozisyonu belirlemek istediğimiz element e positionŞ relative deriz. Ve bnundan sonra top:50 px derken o cismin bulundugu sol en üst hizadan 50px kafasından aşağı inecek yani 50px relative olan elementin kafasıyla, absolute olan elementin kafası arasında mesafe olacak.

Position: fixed = cisim oldugu yerde sabit kalır. Ne kadar aşağıda insen sayfanın ortasında ise normal konumu orada sabit kalır.

Position: sticky = Burada ise cisimi sticky yaptıktan sonra top: 0 diyerek sayfanın 0. Noktasına en üstteki noktası değince cisimin orada sabit kalır her zaman. Yan menüler için top10 tarzı bu özellik kullanılır.

Display: none => Elementi tamamen siler kaldirir html sayfasindan.

Visibility: hidden => Gorunmez yapar elementi ama halen ordadir ve yer kaplamaya devam eder.

**RESPONSIVE ICIN ONEMLI NOKTALAR**

* Viewport veya Scale Ayarla
* Media Queries – Diffrent css styling kullan farklı ekran boyutları için
* PX yerine rem kullan
* Maxwidth ve min width gibi dferğerler kullan
* Ilk önce mobile design yap sonra webe responsive yap

**MEDIA QUERIES**

        body {

            font-family: Arial, Helvetica, sans-serif;

            background: gray;

            color: #fff;

            text-align: center;

            padding-top: 100px;

        }

        h1 {

            display: none;

        }

        /\* Smartphones \*/

        /\* Width 500px e kadar kirmizi, 500px den fazlasinda normal griye doner \*/

        @media (max-width: 500px) {

            body {

                background: red;

            }

            #smartphone h1 {

                display: block;

            }

        }

        /\* Burada ise 500px e kadar mavi, 500px altinda ise tekrar ana degeri griye doner \*/

        @media (min-width: 500px) {

            body {

                background: blue;

            }

        }

<!-- Bu sekilde kullanimdada max width 768 px e kadar yani 0-768 px arasi bu css 'i kullan demis oluyoruz. -->

    <link rel="stylesheet" media="screen and (max-width: 768px)" href="mobile.css">

<style>

        body {

            font-family: Arial, Helvetica, sans-serif;

            background: gray;

            color: #fff;

            text-align: center;

            padding-top: 100px;

        }

        h1 {

            display: none;

        }

        /\*

        Burada ise 500px e kadar mavi, 500px altinda ise tekrar ana degeri griye doner

        @media (min-width: 500px) {

            body {

                background: blue;

            }

        }

        \*/

        /\* Smartphones \*/

        /\* Width 500px e kadar kirmizi, 500px den fazlasinda normal griye doner \*/

        @media (max-width: 500px) {

            body {

                background: red;

            }

            #smartphone h1 {

                display: block;

            }

        }

        /\* Tablet \*/

        /\*

        Sadece max-width 768px yaparsak 0'dan 768px 'e kadar olan herseyi mavi yapar ve yukarda ki 500px e kadar mobile cihazlarda kirmizi olsun istegimiz bosa gitmis olur.

        Bunun olmamasi icin ayriyeten birde min-width 501px yapiyoruz ve diyoruz ki

        max-width 768px 'e  kadar min width de 501 px e kadar asagidaki ozelliklere sahip ol.

        \*/

        @media (min-width: 501px) and (max-width: 768px) {

            body {

                background: blue;

            }

            #tablet h1 {

                display: block;

            }

        }

        /\* Normal \*/

        @media (min-width: 769px) and (max-width: 1200px) {

            body {

                background: green;

            }

            #normal h1 {

                display: block;

            }

        }

        /\* Widescreen \*/

        @media (min-width: 1201px) {

            body {

                background: black;

            }

            #widescreen h1 {

                display: block;

            }

        }

        /\* Landscape \*/

        /\*

        Telefon ekranlari yan cevrildiginde ekran yuksekligi dusecegi icin burada onu yakalayip ona gore bir sayfa design ediyoruz.

        \*/

        @media (max-height: 500px) {

            body {

                background: orange;

            }

            #landscape h1 {

                display: block;

            }

        }

    </style>

**EM and REM**

Em her zaman parent inin ozelliklerini devalir. REM ise her zaman base olarak ayarlanmis degeri baz alarak deger alir.

Bu yuzden REM kullanmak bizim icin cogu zaman daha mantikli ve kullanisli olacaktir.

REM ‘de font size ‘i html tagi altinda bir kere degistirdikten sonra artik her yerde o degeri baz alarak 1,2,3,4,5 rem gibi yani kati seklinde kullanabiliriz. Yani html tagi altinda fontsize ‘ i 10px yaptiysak bundan sonra her yerde 2rem diyerek 2 kati olan 20px diye, 3rem diyerek 3 kati olan 30px gibi ayarlamalar yapabiliriz.

EM’de ise bu durum her zaman parent element’de en son fonsize kac olarak kullanilmissa o deger base deger olarak kabul edilir. Buda bizim isimize cogu zaman gelmez cunku cogu zaman kafa karistiricidir.. Cunku en son nerede em in kac oldugunu yanlis hesaplayabiliriz.

REM ‘de font-siz genellikle 10px yada 62.5% olarak ayarlaniliyor.

**VW and VH (viewport height and viewport width)**

Vh ile heigh verdigimiz element gorunen alanin verdigimiz degeri kadar alan kaplar.

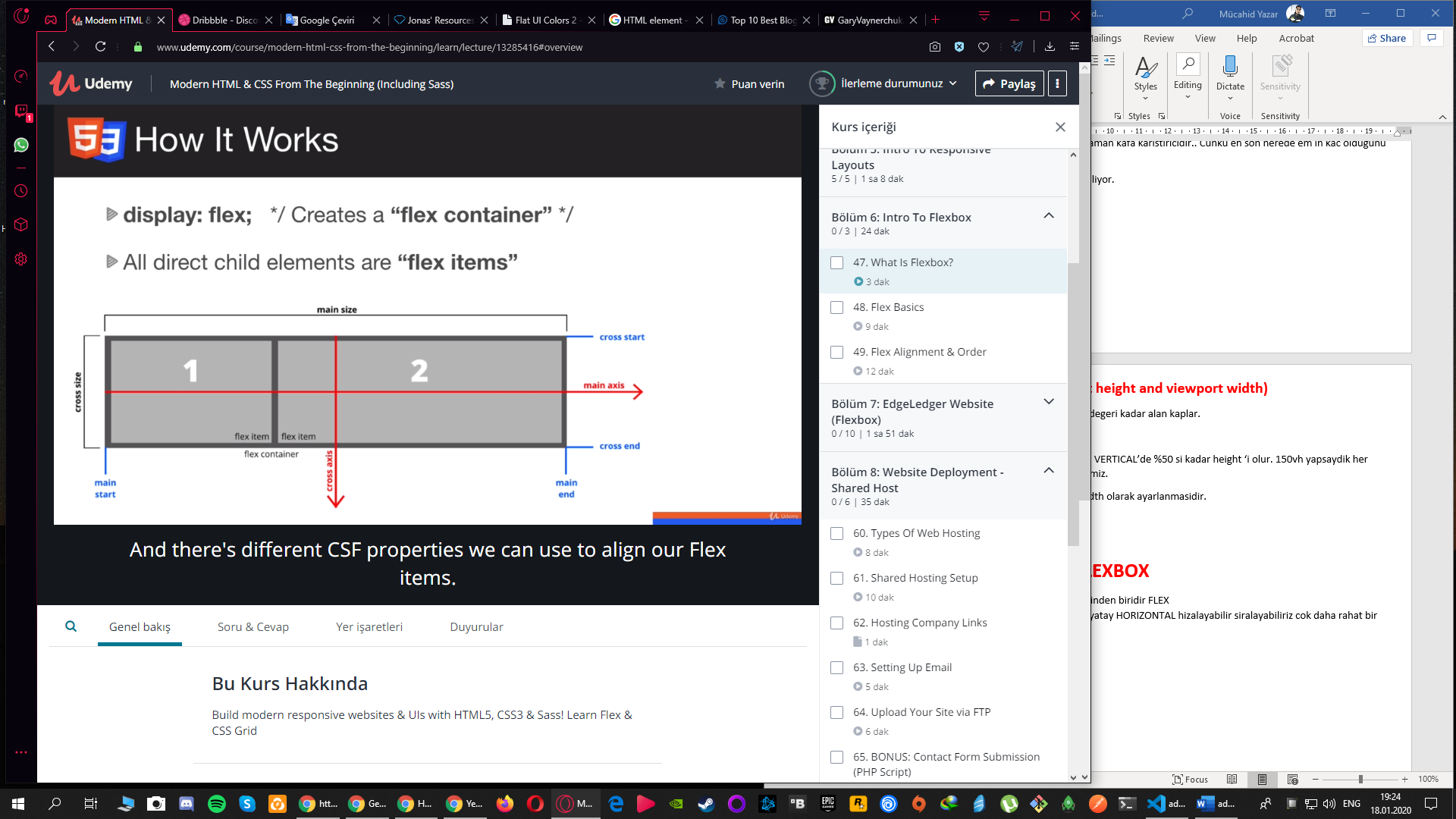
EXAMPLE : header: 50vh

Dersek her zaman monitorde gorunen alanin DIKEY’de yani VERTICAL’de %50 si kadar height ‘i olur. 150vh yapsaydik her zaman gorunen alanin 150% si kadar alani kaplardi height imiz.

Vw ise yine aynisidir aslinda vh nin. Sadece yatay olarak width olarak ayarlanmasidir.

**FLEXBOX**

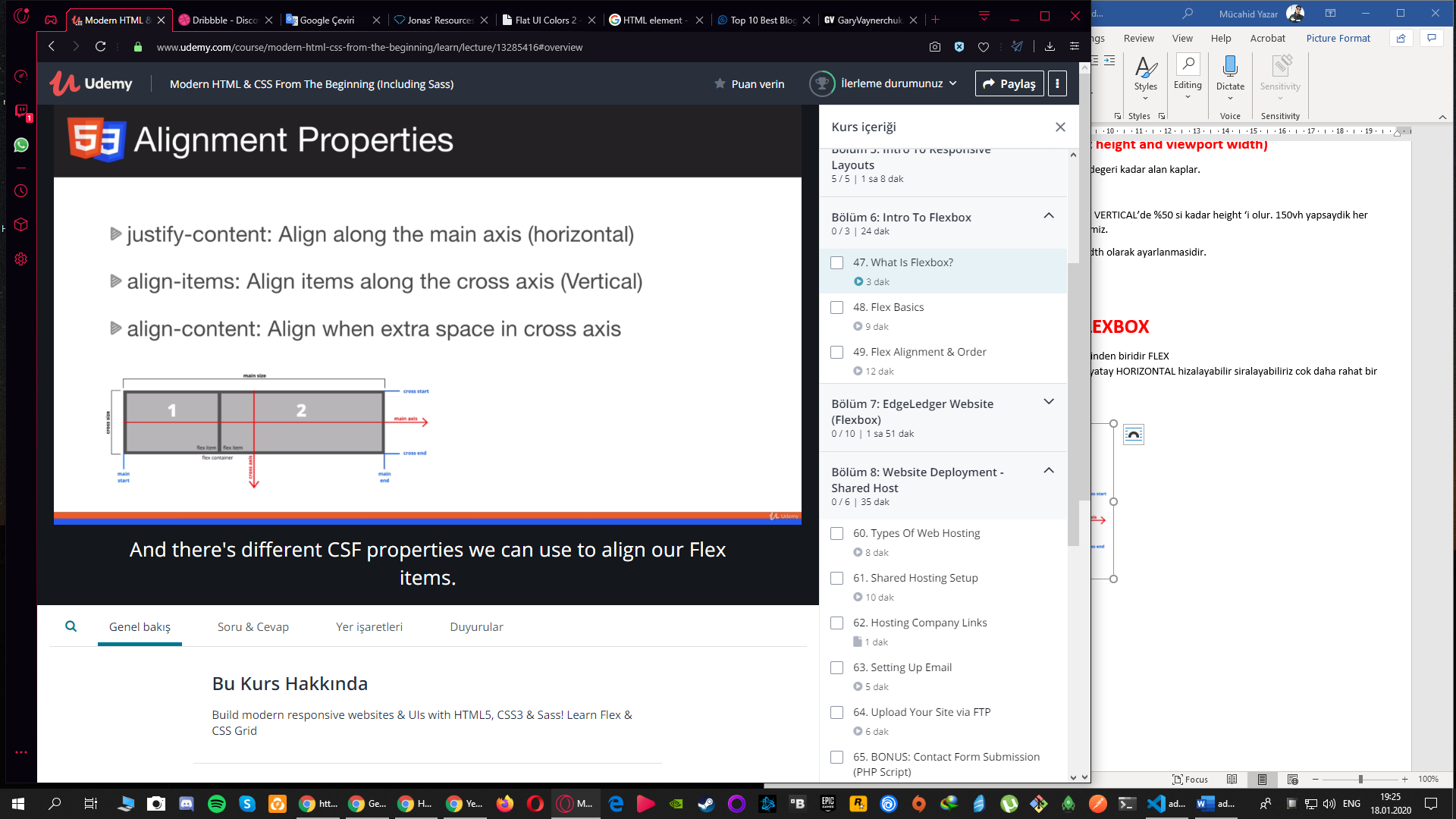
* Float lari birak ve Flex e gec. Modern CSS cozumlerinden biridir FLEX
* Artik itemleri FLEX ile hem dikey VERTICAL hemde yatay HORIZONTAL hizalayabilir siralayabiliriz cok daha rahat bir sekilde



Justify-content = main axis boyunca siralamamiza yarar

Align-item = cross axis boyunca siralamamiza yarar

Align-content = Cross Axisde extra alan oldugunda hizalamamiza yarar



**FLEX BASIC KULLANIMI**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Flex Basic</title>

    <style>

        #container {

            display: flex;

            /\* Dikey olarak siralar elementlerini \*/

            /\* flex-direction: column; \*/

            /\* Default olarak flex yazilinca, flex direction u row kabul eder ve ayni satirda sola hizalar tum elementleri \*/

            /\* Fakat flex direction row-reverse ise saga hizalar tum elementleri ayni satirda\*/

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

            /\* Bu sekilde satirda cok fazla yani 12 den fazla eleman oldugunda scrool bar cikmasini onleriz ve scrool bar cikartmaz bize fazla elemanlari alt satira gonderir. \*/

            /\* Alt satira tarayici ekran boyuttuna gore responsive olarak elemanalri gonderir. Yani ilk 2 gonderir 10 kalir biraz daha kuculur 2 taha gonderir 8 kalir ilk satida gibi... \*/

            flex-wrap: wrap;

            /\* Bu ise flex-wrap: wrap; ile birebir ayni islemi yapar \*/

            /\* flex-flow: row wrap; \*/

            /\* Hersey tek satirda kalmaya devam eder. Birnevi flex-wrap: wrap; yazilmadigini dusunun. \*/

            /\* flex-flow: row nowrap; \*/

        }

        .item {

            background: #f4f4f4;

            border: #ccc solid 1px;

            padding: 1rem;

            text-align: center;

            /\* flex: 1 diyerek tum elemanlara ayni satirda esit genislik vererek tum satiri kaplamalarini saglar. \*/

            flex: 1;

            /\* Margin verirsek floattaki gibi tasma dis alan yani kapladiklari alan kuculur fakat margini cevrelerine verirler \*/

            margin: 0 0.75rem;

        }

        .item:first-child {

            /\* Bu sekilde 2 verirsek satirda yarisini kaplamasini soyleriz. \*/

            /\* 3 verirsek %75 ini kaplamasini soyleriz. \*/

            flex: 1;

        }

    </style>

</head>

<body>

    <div id="container">

        <div class="item">

            <h3>Item 1</h3>

        </div>

        <div class="item">

            <h3>Item 2</h3>

        </div>

        <div class="item">

            <h3>Item 3</h3>

        </div>

    </div>

</body>

</html>

**FLEX ALIGNMENT & ORDER**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Flex Align</title>

    <style>

        #container {

            background: #555;

            height: 600px;

            display: flex;

            /\* flex-start => default olarak gelen degerdir ve tum itemlari solda siralar \*/

            /\* flex-end => tum itemleri ayni satirda sagda hizalar. Item1 - Item 2 - Item3 olarak \*/

            /\* center => tum itemlari ortada hizalar \*/

            /\* space-arround => tum bosluklar itemlarin etrafinda esit olarak dagitilir. Evently ile aralarindaki fark ise bosluklar tum itemlere ayri ayri esit dagitildigindan item 1 ve2 arasinda 2 ve 3 arasinda daha fazla bosluk olur. \*/

            /\* space-between => tum bosluklar aralarinda hizalanir. Ilk bastaki itemin solunda ve sonraki itemin saginda bosluk olmaz. \*/

            /\* space-evenly => itemlerin tum taraflarina bosluklar esit dagitilir. \*/

            justify-content: space-evenly;

            /\* justify-content yatayda hizalama yapar\*/

            /\* stretch => itemlarin hepsi containerin uzunlugu kadar asagi boyu uzatilir. Default olarak gelen degerdir. \*/

            /\* flex-start => containerin en ust hizasinda tutar cisimleri ve itemlarin parent elemente gore boyutlarinin uzamasini engeller \*/

            /\* flex-end => Containerin en alt hizasinda tutar cisimleri ve uzamasini engeller \*/

            /\* center => Itemlari diker olarak containerin ortasinda konumlandirir ve kendi boyutlarinda tutar ve stretch uzatmasini yine onler \*/

            /\* baseline => flex-start ile ayni \*/

            align-items: stretch;

            /\* align-content dikeyde hizalama yapar (Yukseklik olmasi lazim) \*/

            /\* container 600px ve bizim flew-wrap ile 2 satira dusurulmus 100px uzunlugunda 12 tane kutumuz var diyelim. 6 tanesi bir 6 tanesi bir \*/

            /\* Fakat normalde 1. 6 kutuyu en ustten siralar 2. 6 kutuyu 300. px yukseklikten siralar \*/

            /\* flex-start => yaparak tum satirlari yani 2 6 kutulu satiri alt alta aralarinda bosluk olmadan siralariz. \*/

            /\* flex-end => yaparak tum satirlari en alta dayali hizalariz \*/

            /\* center => tum satirlari dip dibe aralarinda bosluk olmadan ortada hizalariz \*/

            /\* space-around => 2 kutu grubu arasinda her birine bosluklari esit olarak dagitir \*/

            /\* space-between => default olarak gelen degerdir tum bosluk input gruplari arasina konulur \*/

            /\* space-evenly => item gruplarinin her bir grubuna dikey olarak aralarina esit bosluk konulur. \*/

            align-content: center;

            /\* align-content dikeyde fazla olan bosluklari dagitma islemi yapar \*/

        }

        .item {

            background: #f4f4f4;

            border: #ccc solid 1px;

            text-align: center;

            padding: 1rem;

            margin: 0.5rem;

            /\* width verir. Width ile ayni islevi gorur. \*/

            flex-basis: 200px;

        }

        /\* Order ilede itemlari istedigimiz sira ile konumlandirilmalarini saglayabiliriz. \*/

        .item-1 {

            order: 3;

        }

        .item-2 {

            order: 1;

        }

        .item-3 {

            order: 2;

        }

        .item-2 {

            /\* flex-start => dikeyde en basta konumlandirilir. \*/

            /\* flex-end => dikeyde en dipte konumlandirilir. \*/

            /\* center => dikeyde ortada konumlandirir \*/

            /\* stretch => uzatir \*/

            /\* baseline => asil konumunda konumlandirir. \*/

            align-self: flex-end;

            /\* align-self => Tek bir itemi dikey olarak hizalamak \*/

        }

    </style>

</head>

<body>

    <div id="container">

        <div class="item item-1"><h3>Item 1</h3></div>

        <div class="item item-2"><h3>Item 2</h3></div>

        <div class="item item-3"><h3>Item 3</h3></div>

    </div>

</body>

</html>

**TARGET SELECTORS**

    <style>

      /\* div icindeki tum p leri secer \*/

      div p {

      }

      /\* if div is a direct parent of p \*/

      /\* parent elementi div olan tum p leri kapsar \*/

      div > p {

        background-color: #ddd;

      }

      /\* if p is right after div \*/

      /\* Divden hemen sonraki yani ayni hizadaki p yi secer. Yani next sibling mantigi jsde ki \*/

      div + p {

        background: #333;

        color: #fff;

      }

      /\* if p is preceded by a ul with same parent \*/

      /\* Ayni parent elemente sahip, ul ile ayni parent elemana sahip p yi secer \*/

      ul ~ p {

        background: #ff0000;

        color: #fff;

      }

      /\* Target attribute \*/

      /\* attribute secme yontemi ile atribute si si target olani secme. \*/

      a[target] {

        background: #ddd;

      }

      /\* Target specific attribute value \*/

      /\* Detayli atribute secme yontemi ile atributeleri hem text hemde email olanlari seciyoruz \*/

      input[type='text'],

      input[type='email'] {

        width: 100%;

      }

    </style>

**PSEUDO SELECTORS**

    <style>

      li {

        padding: 0.25rem;

        margin: 0.25rem;

        list-style: none;

      }

      /\* first-child \*/

      li:first-child {

        background: red !important;

      }

      /\* last-child \*/

      li:last-child {

        background: blue;

      }

      /\* nth-child selector matches every element that is the nth child, regardless of type, of its parent. \*/

      /\* Position 3 \*/

      /\* ul icerisinde ki 3. li yi secer \*/

      li:nth-child(3) {

        background: purple;

      }

      /\* Every 3rd after 0 \*/

      /\* Her 3. 0'dan sonra 3er 3 er sayarak \*/

      /\* 3 er 3 er sayarak her 3. secilir. 3. 6. 9. .... liler secilir \*/

      li:nth-child(3n + 0) {

        background: orange;

      }

      /\* Every 3rd after 7 \*/

      /\* Her 3. 7 den baslayarak secilir 7. 10. 13. 16. diye secilerek gider \*/

      li:nth-child(3n + 7) {

        background: yellow;

      }

      /\* odd \*/

      /\* Her tek sayili eleman \*/

      li:nth-child(odd) {

        background: #ccc;

      }

      /\* even \*/

      /\* Her cirf sayi olan eleman \*/

      li:nth-child(even) {

        background: #ddd;

      }

    </style>

**BEFORE and AFTER PSEUDO SELECTORS**

<style>

      body {

        font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;

        background: #333;

        color: #fff;

      }

      /\* is-required classi alan elementden sonrasina ekleme yapar. Content ile icerigine ekleme yapariz. \*/

      /\* Ve eklenen tum styling lerde sonradan eklenen kisim icin yapilir. \*/

      .is-required::after {

        content: '\*';

        color: red;

        padding-left: 2px;

      }

* Asagida ise header ve onun arkasina before ile opcity verilmsi background koyduk
* Dikkat edilmesi gerekenler
  + ::before yyaptiktan sonra mutlaka bir content eklemeliyiz yoksa sayfada gosteremeyiz.
  + ::before dan sorna bu gibi islemlerde position vermeliyiz yoksa yine hersey duzgun olsada sayfada goremeyiz backgroundumuzu
  + ::before yaptiktan sonra background verdiyse eger header daha uste kalmasi gerekiyor ve bunun icin background koydugumuz before nin zindex ini -1 veya daha da fazla isteginize gore dusurun

      header {

        display: flex;

        flex-direction: column;

        justify-content: center;

        align-items: center;

        text-align: center;

        height: 100vh;

      }

      header h1 {

        font-size: 6rem;

        margin: 1rem;

      }

      header::before {

        content: '';

        background: url('https://source.unsplash.com/1600x900/?nature,water')

          no-repeat center center/cover;

        position: absolute;

        top: 0;

        left: 0;

        width: 100%;

        height: 100%;

        z-index: -1;

        opacity: 0.5;

      }

    </style>

**BOX SHADOW**

    <style>

        /\* offset-x | offset-y | color \*/

        box-shadow: 10px 10px teal;

        /\* offset-x | offset-y | blur-radius | color \*/

        box-shadow: 5px 5px 20px teal;

        /\* Negative values \*/

        box-shadow: -5px -5px 20px teal;

        /\* offset-x | offset-y | blur-radius | (yayilma)spread-radius | color \*/

        box-shadow: 3px 3px 10px 1px rgba(0, 0, 0, 0.3);

        /\* inset | offset-x | offset-y | color \*/

        /\* insette verilen degerleri ters kabul eder 3 ise -3, -4 ise +4 kabul eder \*/

        box-shadow: inset -3px -3px teal;

        /\* Multiple shadows \*/

        /\* Cift golgelendirme yapma \*/

        box-shadow: 3px 3px teal, -3px -3px olive;

      }

    </style>

**TEXT SHADOW**

    <style>

      h1 {

        font-family: Arial, Helvetica, sans-serif;

        font-size: 5rem;

      }

      h1.a {

        /\* h-shadow | v-shadow | color \*/

        text-shadow: 0.2rem 0.2rem steelblue;

      }

      h1.b {

        /\* h-shadow | v-shadow | blur \*/

        text-shadow: 0.4rem 0.3rem 0.7rem steelblue;

      }

      h1.c {

        /\* White Text \*/

        color: #fff;

        text-shadow: 0.2rem 0.2rem 1rem steelblue;

      }

      h1.d {

        /\* Negetive Values \*/

        text-shadow: -0.4rem -0.3rem 0.7rem steelblue;

      }

    </style>

**CSS VARIABLES ( CUSTOM PROPERTIES )**

* Variables larla degiskenler belirleyip css lerde sonra bunlari belirledigimiz hedef deki yerler var ile cagirip parantez icinde kullanabiliriz.
* Asagida header icin variables lar belirledik ve daha sorna bunlari ayni sayfada header icin var ile cagirip kullandik

      header {

        --header-bg-color: #333;

        --header-txt-color: #fff;

        --header-txt-align: center;

      }

      header {

        /\* --header-bg-color: green; \*/

        background-color: var(--header-bg-color);

        color: var(--header-txt-color);

        text-align: var(--header-txt-align);

      }

* Yada asagida ki gibi :root scope una tanimlariz kullanacagimiz tum degerleri ve daha sonra bunlari istedigimiz her yerde var ile cagirabiliriz.

:root {

        --container-max: 960px;

        --box-bg-color: steelblue;

        --box-txt-color: #fff;

        --box-1-width: 1;

        --box-2-width: 3;

        --header-bg-color: #333;

        --header-txt-color: #fff;

        --header-txt-align: center;

      }

**Keyframe Animation**

    <style>

      body {

        background: #333;

      }

      .box {

          position: relative;

          background: #fff;

          width: 200px;

          height: 200px;

          /\* Bu sekidle animasyonu isimlendirerek yada tanimlayarak asagidaki gibi cagirabiliriz \*/

          animation-name: animate1;

          /\* Animasyonun 1 turunun kac saniye icinde tamamlanacagi \*/

          animation-duration: 3s;

          /\* Animasyon tekrar sayisi - Sinirsiz yapmak icin => infinite => yazin \*/

          animation-iteration-count: 2;

          /\* Bitis noktasinda kalmasini soyleriz en son tur bitince \*/

          animation-fill-mode: forwards;

          /\* Animation un sayfa acilinca ne kadar gecikme sonrasi calisacagini belirleriz \*/

          animation-delay: 1s;

          /\* animation-direction: alternate => from dan to ya animate eder \*/

          /\* animation-direction: alternate-reverse => to dan from a animate edilmesini saglar \*/

          animation-direction: alternate-reverse;

          /\* ease in animation yavas baslar hizlanir gibi anlamlari vardir \*/

          /\* Bunun gibi cok fonksiyon vardir burada detaylari internetten bakin \*/

          animation-timing-function: ease-in;

          /\* Yada yukaridakilerin hepsini sileriz ve asagidaki gibi tek satirda kullanabiliriz \*/

          /\* @keyframes duration | timing-function | delay | iteration-count | direction | fill-mode | play-state | name \*/

          animation: animate1 5s forwards 1s ease-in-out;

      }

      @keyframes animate1 {

          /\* Nereden : Baslangic ozelliklerini belirleriz animation un \*/

          from {

              top: 0;

              width: 100px;

          }

          /\* Nereye : Bitis ozelliklerini yani turu nerede tamamlayacagini belirleriz buradada \*/

          to {

              top: 500px;

              width: 600px;

              background-color: green;

          }

      }

    </style>

**CSS Transition**

    <style>

      body {

        background: #333;

        display: flex;

        align-items: center;

        justify-content: center;

        height: 100vh;

      }

      .box {

          background: #fff;

          width: 100px;

          height: 100px;

          /\* Hangi propertylere transition uygulansini buradan seciyoruz. Hepsi icin => all => yazmamiz yeterli oluyor. \*/

          transition-property: background;

          /\* transition isleminin suresini ne kadar surede olacagini s cinsinden yaziyoruz \*/

          transition-duration: 2s;

          /\* transition hizlari ile fonksiyonlari ayarlariz \*/

          transition-timing-function: ease-in;

          /\* Gecikmeyi ayarlariz \*/

          transition-delay: 1s;

          /\* yada hepsini asagidaki sirasiyla tek satirdada ayarlayabiliriz. \*/

          /\* property name | duration | timing function | delay \*/

          transition: background 2s ease-in-out;

          /\* Yada yukaridaki yerine asagidaki gibi virgulle beraber ikili uclu veya daha coklu kullanimda yapabiliriz. \*/

          transition: background, border-radius 2s ease-in-out;

          /\* all \*/

          transition: all 2s ease-in-out;

      }

      .box:hover {

          background: red;

          border-radius: 50%;

          width: 300px;

          height: 300px;

      }

    </style>

**ALL TRANSITION PROPERTIES**

/\*

        TRANSITIONAL PROPERTIES

        -Properties that have an identifiable halfway point

        background-color

        background-position

        border-color

        border-width

        border-spacing

        bottom

        color

        font-size

        font-weight

        height left

        letter-spacing

        line-height

        margin

        max-height

        max-width

        min-height

        min-width

        opacity

        outline-color

        outline-offset

        outline-width

        padding right

        text-indent

        text-shadow

        top

        vertical-align

        visibility

        width

        word-spacing

        z-index

\*/

**TRANSFORM and TRANSLATE**

    <style>

      body {

        background: #333;

        display: flex;

        align-items: center;

        justify-content: center;

        height: 100vh;

      }

      .box {

        background: white;

        width: 300px;

        height: 300px;

        /\* Transform - rotate, scale, skew \*/

        /\* 25 derece saga eger \*/

        /\* transform: rotate(25deg); \*/

        /\* 25degree lik paralellik olsuturur sol ve sagi \*/

        /\* transform: skew(25deg); \*/

        /\* 2 yazarsa karesi kadar 3 yazarsak kupu kadar buyutur Scale eder 3.5.6.7. diye devam edebilir \*/

        /\* transform: scale(2); \*/

        transition: all 1s ease-in-out;

      }

      .box:hover {

        /\* transform: rotate(180deg); \*/

        /\* transform: skew(25deg); \*/

        /\* transform: scale(2); \*/

        /\* border-radius: 50%;

        background-color: blue; \*/

        /\* translateY ile oldugu konumdan icindeki deger kadar asagi yada yukari gondeririz.

        Pozitif sayi ile asagi negatif sayi ile yukari hareket ettiririz. \*/

        /\* transform: translateY(100px); \*/

        /\* translateX pozitif sayi ile saga, negatif sayi ile sola hareket ettiririz. \*/

        /\* transform: translateX(-100px); \*/

        /\* x & y birlikte kullanimi \*/

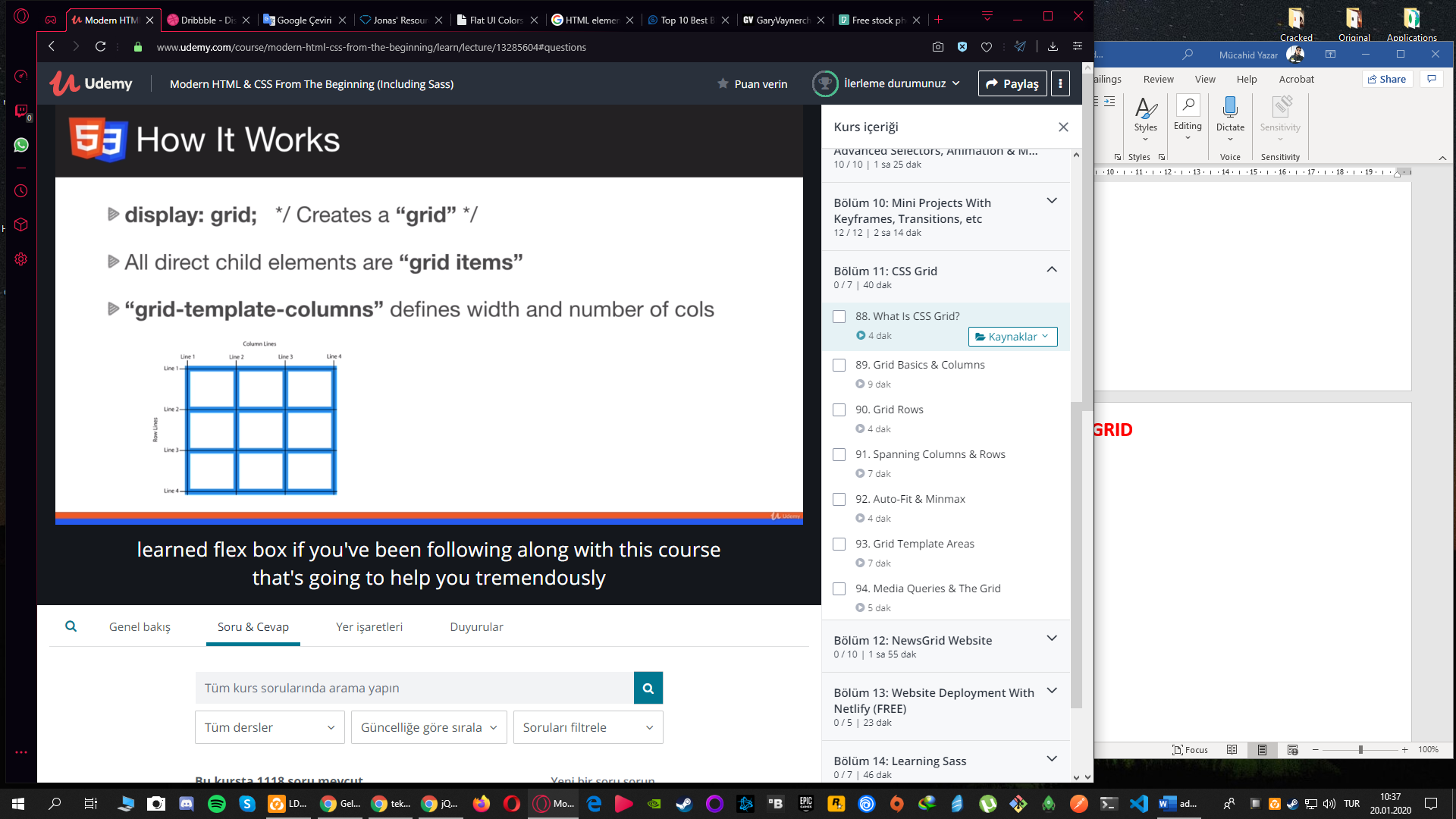
        transform: translate(100px, 100px);

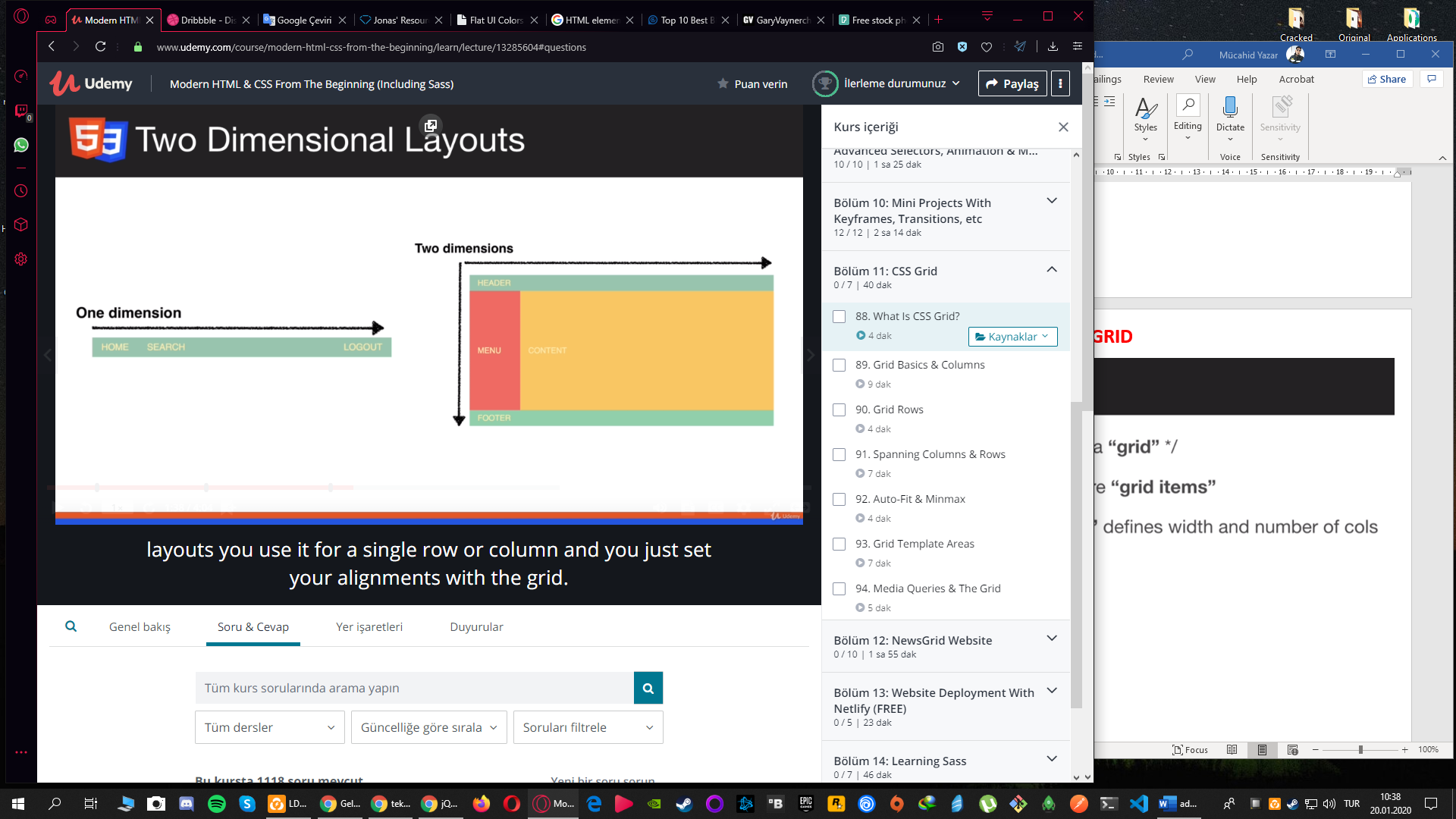
        transform: translate3d(100px, 100px, 100px);

      }

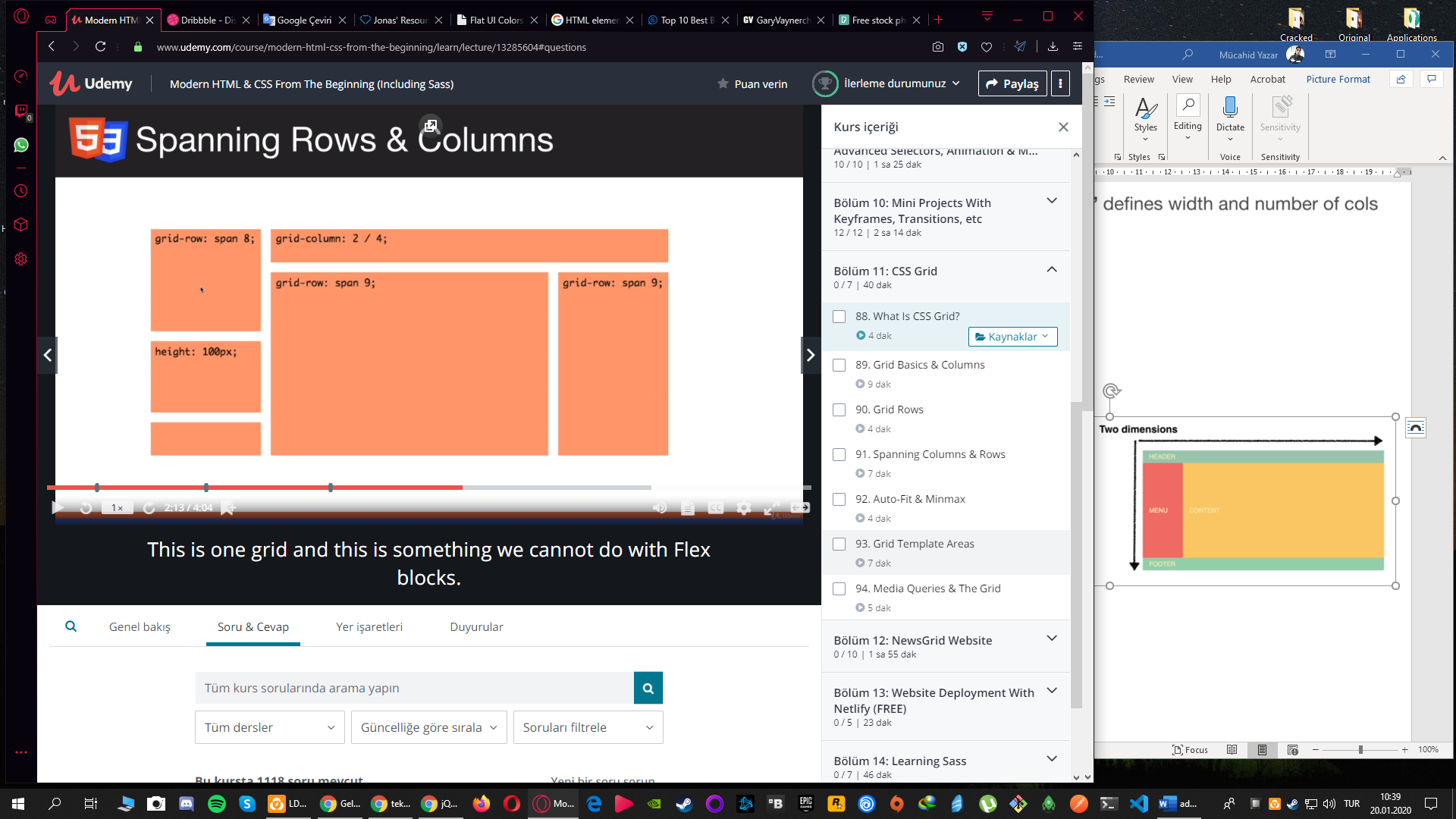
    </style>

**GRID**





GRID ORNEK



GRID vs FLEX

* Aslında ıkısıde bır bırının yerını tutamaz bazen GRİD bazen FLEX kullanmamız gerekiyor
* Düzenleme ve sistemleri ve daha dıştaki elemanları grid ile düzenle
* Basit hizalamaları daha içerideki elemanları flex ile düzenle Menu items ları gibi
* Genellikle flex i tercih ederim ben ama hepsini kullanacagız

GRIDS

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>CSS GRID</title>

    <style>

        .grid {

            /\* Ilk bas flexbox daki gibi parent elementin display ine grid deriz. Fakat sadece grid yapmayla birsey olmaz. \*/

            display: grid;

            /\* Burada ne kadar olculendirme yaparsak grid o kadar parcaya bolunecek. Ve sirasiyla icindeki elemanlar bu parcalara yerlesecekler. \*/

            /\* Eger asagidaki degerlerden birisini auto yaparsak ornegin birincisini sayfa uzadiginda digerleri verilmis olcude kalirken auto verdigimiz 1. si ise sayfayi dolduracak sekilde uzayacaktir. \*/

            /\* Bu ayarlama her satir icin gecerlidir \*/

            grid-template-columns: auto 200px 200px 100px;

            grid-template-columns: 200px 200px 200px;

            /\* Veya yukaridakinnin aynisini asagidaki sekildede yapabiliriz \*/

            grid-template-columns: repeat(3, 200px);

            /\* Veya asagidaki sekilde yuzdelik bolumlemeler yapabiliriz. \*/

            /\* satirin tamami toplam fr sayisidir ve verilen fr lara gore bolunur \*/

            grid-template-columns: 1fr 2fr 1fr;

            /\* Tum itemlarin cevresine AYRI AYRI 1rem bosluk verir \*/

            grid-gap: 1rem;

        }

        .item {

            padding: 3rem;

            border: #ccc 1px solid;

            background: #f4f4f4;

            font-size: 1.3rem;

            font-weight: bold;

            text-align: center;

        }

    </style>

</head>

<body>

    <div class="grid">

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

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

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

        <div class="item">item 4</div>

    </div>

</body>

</html>

ROWS

.grid {

            /\* Ilk bas flexbox daki gibi parent elementin display ine grid deriz. Fakat sadece grid yapmayla birsey olmaz. \*/

            display: grid;

            /\* Burada ise 6 itemımız olugu halde 3 tane fr verdık \*/

            /\* 1fr item ın default özelliğidir. 2fr ise itemin default height değerinin 2 katıdır. \*/

            grid-template-rows: 1fr 2fr 3fr;

            /\* Eger asagıdakı gıbı yaparsakda yuakrıda yaptıklarımız ilk 3 ü için uygulanır ve kalanları için aşşağıda verdiğimiz değer uygulanır. \*/

            grid-auto-rows: 3fr;

            /\* Bu sekılde ıkı ozellıgı bırde kullanabılırız \*/

            grid-template-columns: repeat(2, auto);

        }

SPANNING COLUMN and ROWS

        .item:first-child {

            /\* Bu sekilde yaparsak satirin ilk column p[arcasi bos olur ve column 2. siradan baslar ve devam eder. Default olarak 1 den baslar \*/

            grid-column-start: 2;

            /\* Asagidaki gibi ayarlarsa yukarisi iptal olur ve firstchild 3 columnluk yer kaplar  \*/

            grid-column-start: 1;

            grid-column-end: 4;

            /\* column start end in kisa yolu \*/

            /\* Ilk deger yani 1 nereden baslasin kacinci columndan baslasin demektir \*/

            /\* Burada ise 1 fazla deger yazmamiza gerek yok kac column istiyorsak spanden sonra onu yazacagiz \*/

            grid-column: 1 / span 3;

            /\* Bu seferde grid 1. satiri alir ve 2. satiri alir. 3. yu almaz ve 3. nun sinirda biter \*/

            grid-row-start: 1;

            grid-row-end: 3;

            /\* row start end in kisa yolu \*/

            /\* Ilk deger yani 1 nereden baslasin kacinci columndan baslasin demektir \*/

            /\* Burada ise 1 fazla deger yazmamiza gerek yok kac column istiyorsak spanden sonra onu yazacagiz \*/

            grid-row: 1 / span 2;

        }

**MINMAX**

.grid {

            display: grid;

            /\* auto-fit ile tum satiri auto kaplasin diyoruz \*/

            /\* minmax ilede ilk once minimum ne kadar genislikte olacagini ve sonrada maximum ne kadar genislikte olacagini belirliyoruz \*/

            /\* yukariyi yaptiktan sonra sayfa genisligine gore columnlar satirda artacak veya azalacaktir. \*/

            grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));

        }

**GRID-AREAS**

        .container {

            display: grid;

/\* Bu sekilde ornek template alani hazirliyoruz ve daha sonra bunlari grid-are ile sectiklerimize atayacagiz\*/

/\* ilk 3 columnu sadece header a atiyoruz. Eger 1 tane grid-area header tanimlarsak tanimlanan element tum satiri kaplar. 2 tane tanimalrsa 2ye bolerek paylasirlar. 3 tane tanimalrsak 3e bolerek paylasirlar. \*/

/\* ikinci satirin ise 2 tanesini content olarak 1 tanesini sidebar olarak ayarliyoruz \*/

/\* Hep 3 e bolebiliriz ve isimler onemli degil sitediginzi gibi cagirabilrisiniz \*/

/\* Ayarlamalar bittikten sonra grid-area lar ayarlanana kadar icerikler otomatikmen sirayla alanlara yerlesir. \*/

            grid-template-areas:

                'header header header'

                'content content sidebar'

                'box-1 box-2 box-2'

                'box-3 box-3 box-3'

                'footer footer footer';

            grid-gap: 1rem;

        }

        .header {

            /\* Burada ise .header elementine grid-area nin grid sistemindeki header alanini tahsil ettik \*/

            grid-area: header;

            text-align: center;

        }

        .content {

            grid-area: content;

        }

        .sidebar {

            grid-area: sidebar;

        }

        .box-1 {

            grid-area: box-1;

        }

        .box-2 {

            grid-area: box-2;

        }

        .box-3 {

            grid-area: box-3;

        }

        .footer {

            grid-area: footer;

            text-align: center;

        }

GRID-AREA and MEDIA QUERY

@media (max-width: 500px) {

            .container {

                grid-template-areas:

                'header'

                'content'

                'sidebar'

                'box-1'

                'box-2'

                'box-3'

                'footer';

            }

        }

GRID and MEDIA QUERY

        .grid {

            display: grid;

            /\* her satir 4 parca olsun auto strech yapip satiri kapslasin \*/

            grid-template-columns: repeat(4, auto);

            grid-gap: 1rem;

        }

        @media (max-width: 768px) {

            .grid {

                /\* her satir 2 parca olsun auto strech yapip satiri kapslasin \*/

                grid-template-columns: repeat(2, auto);

            }

        }

        @media (max-width: 500px) {

            .grid {

                /\* her satir tek parca olsun \*/

                grid-template-columns: 1fr;

            }

        }

**HINTS**

HTML ve CSS için basic kullanımlar için underground içine göz at

İd ler sadece bir kere kullanılır html sayfasında, class lar ise istediğimiz kadar kullanılır.

CELAR: BOTH = Float left yaptıktan sonra ortaya cıkan karısıklıkları clear: both tanımlanmıs bir css li element yani divi float left kullanılan elemaentten sonra yazarak duzeltırız.

:HOVER = ÜSTÜNE GELINCE oluşacak efekt

:ACTIVE = Tıklanınca oluşacak efekt

:LAST-CHILD = CHILD elemenı olan elementlerin son çocugunu seçer.