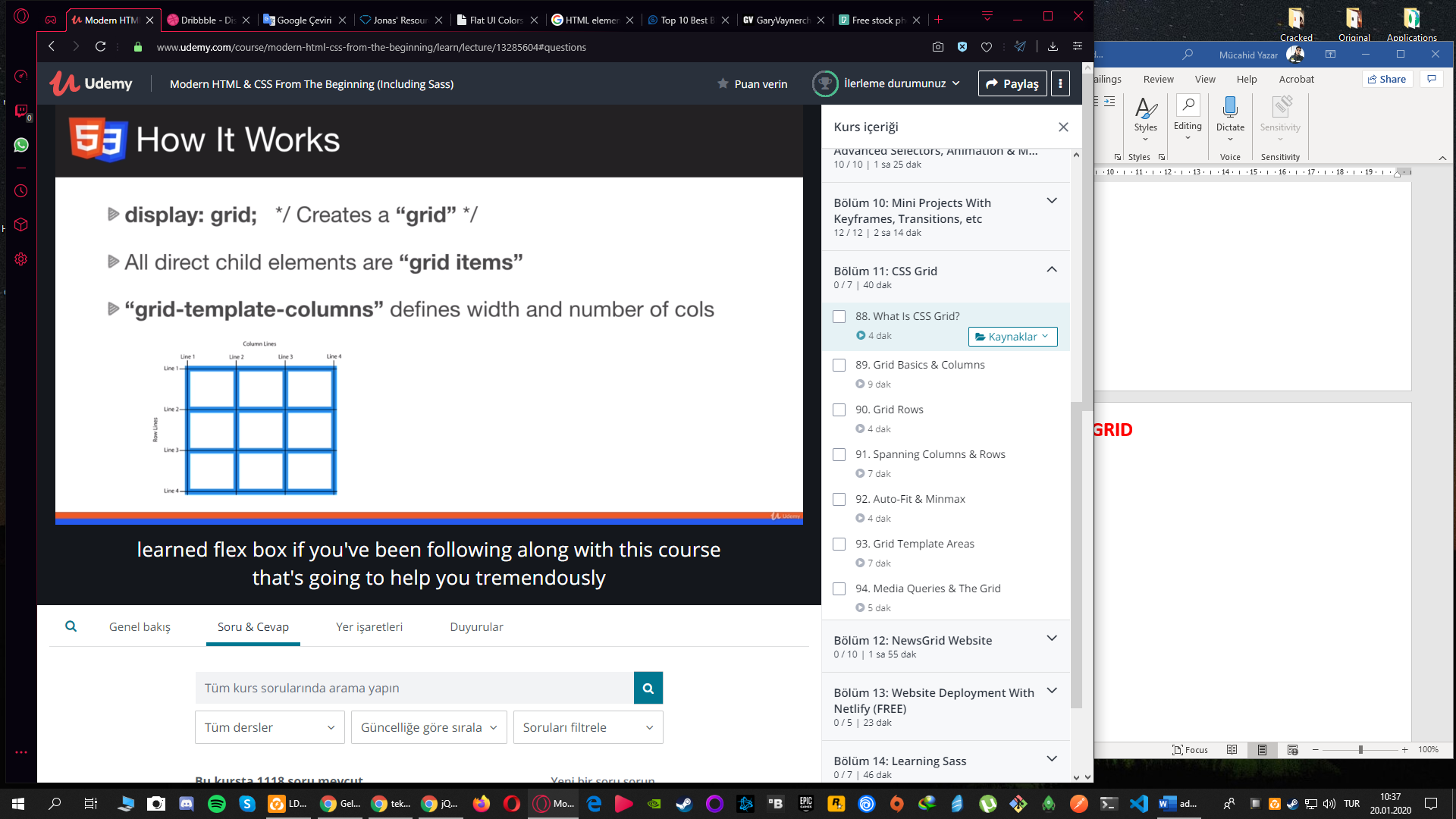
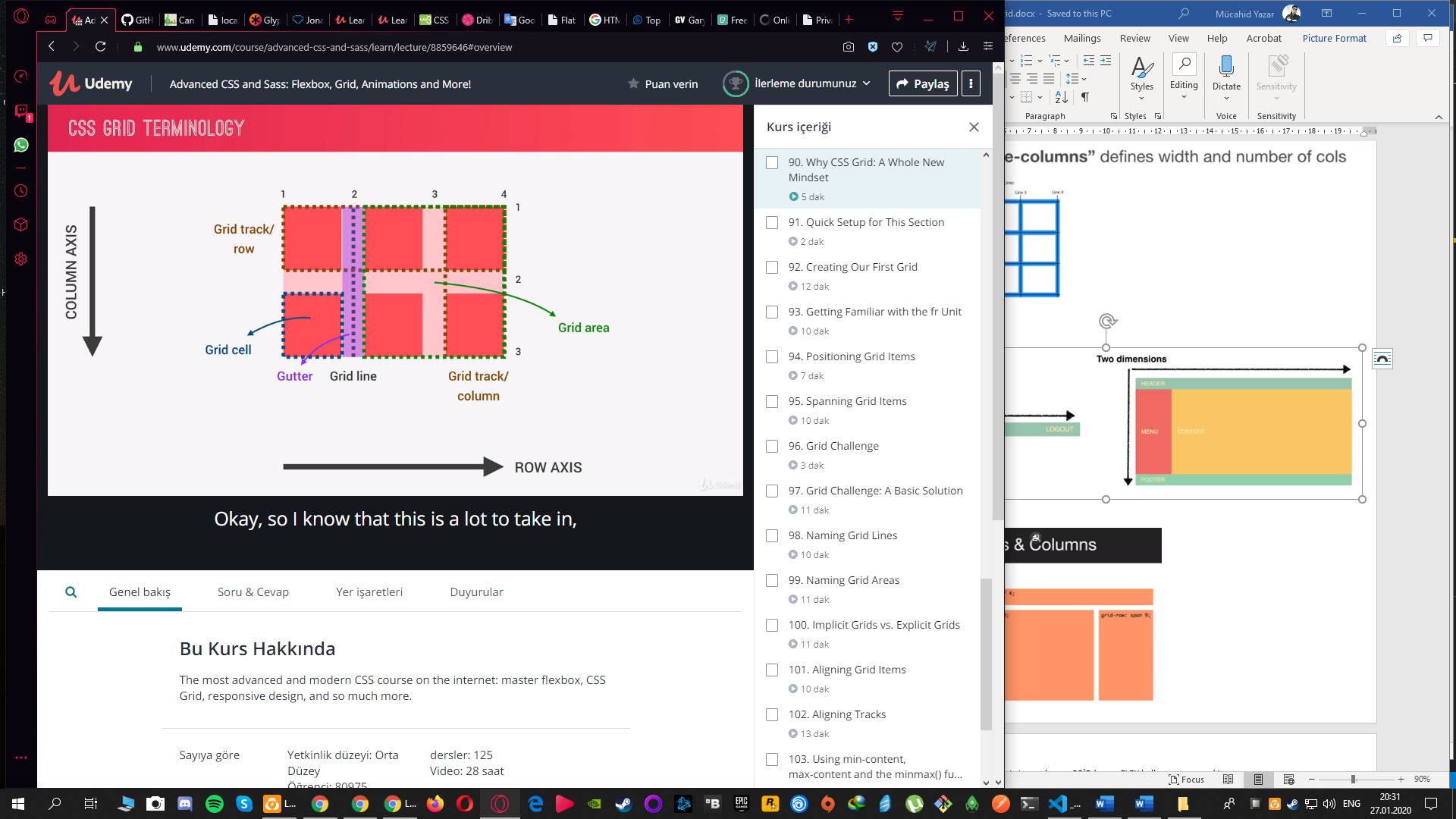
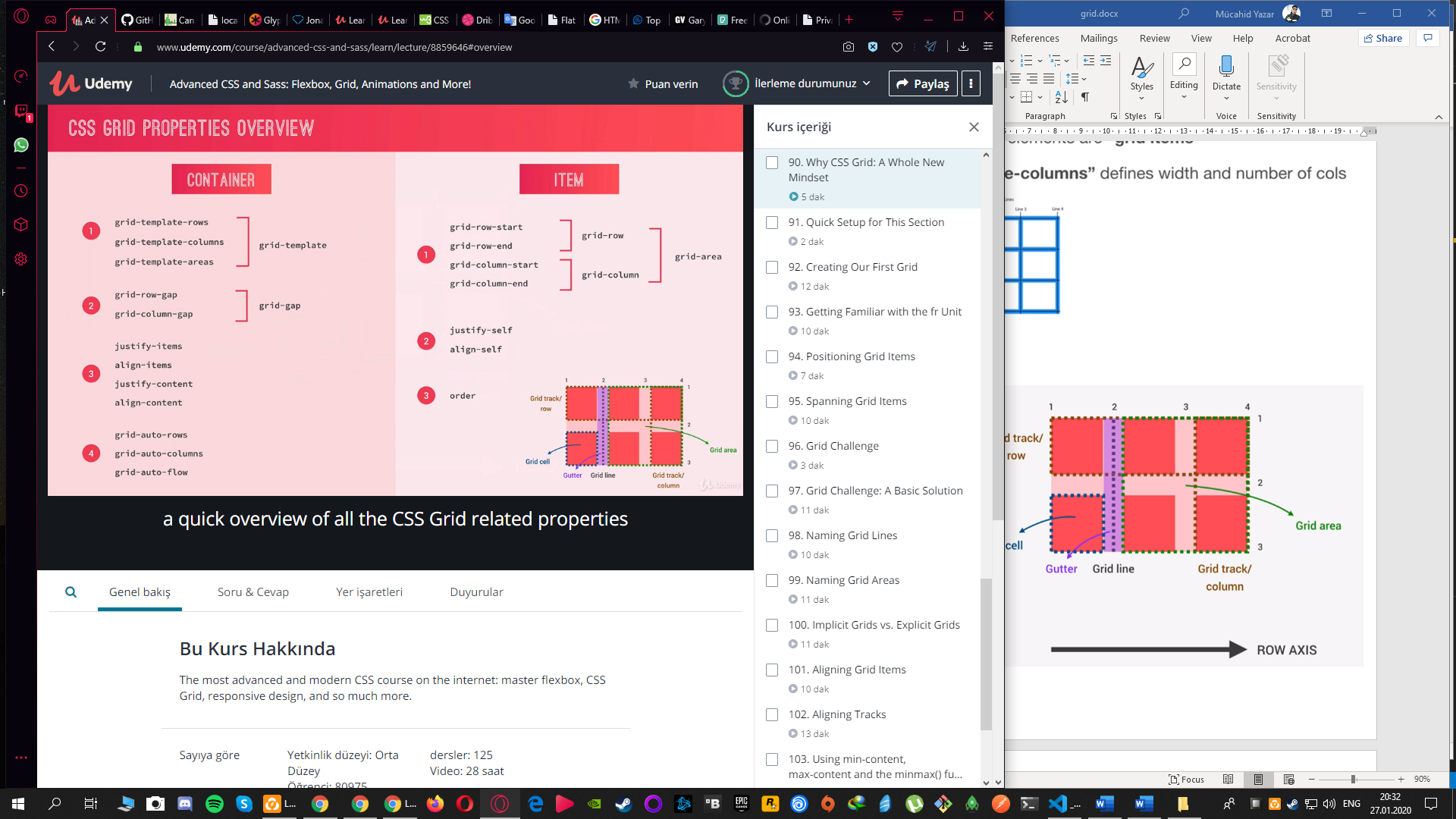
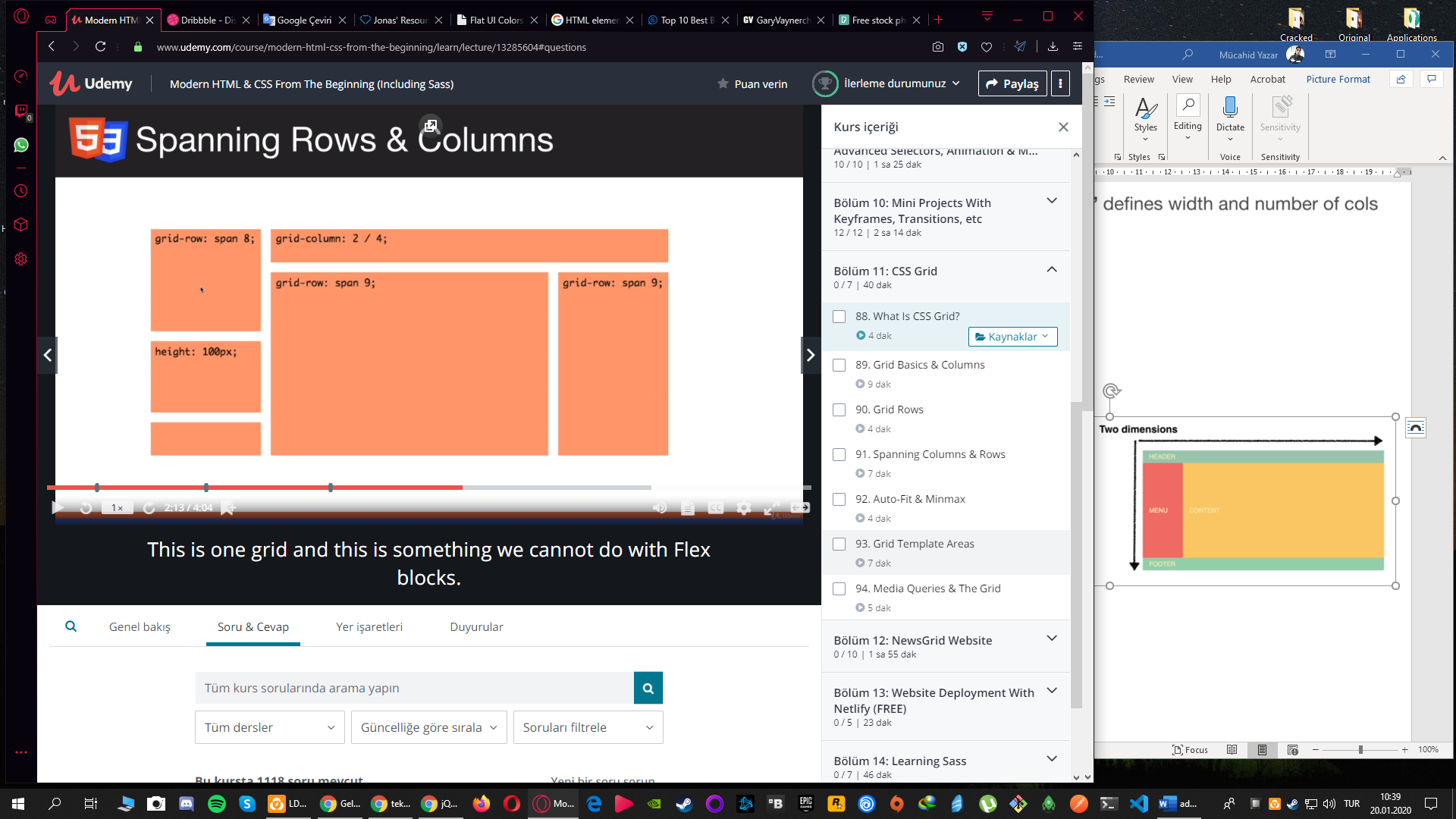
**GRID**



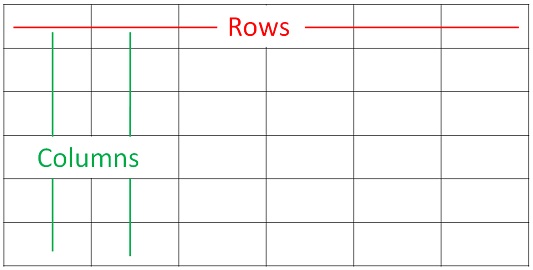


GRID ORNEK



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;

ayri olara grid-row-gap ve grid-column-gap kullanimlarida vardir.

        }

        .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 {

            display: grid;

Ilk rowdaki yani satirdaki yani yatay hizadaki itemlerin yuksekligi 150px olur

2. rowdaki itemlerin yuksekligi 200px olur.

3. ve sonrasindaki rowdaki itemlerin yuksekligi sadece satir yuksekligi olur.

Eger 3 eleman olur ve bunlarin 3u ilk satirda hizalansa bile 2. Satir 200px olarak row olarak acilir bosluk olusturulur.

grid-template-rows: 150px 200px;

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

            /\* 1fr item ın default özelliğidir. 2fr ise itemin default height değerinin 2 katıdır. \*/ //Burada ise 1fr 1satir yuksekligidir. 2fr bir satir yuksekliginin 2 katidir.

            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;

        }

//Buradaki only screen and istege bagli eklenebilir.

//Bunun sayesinde birisi sayfamizdan cikti almak istersek css buna izin vermez

        @media only screen and (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;

            }

        }

GRID TEMPLATE => grid-template-columns and grid-template-row u birlikte kullanabiliriz bu sekilde.

First Rows / Second Columns

.container {

  grid-template-columns: 5px min-content 5px;

  grid-template-rows: 20px min-content 20px;

  grid-template: 20px min-content 20px / 5px min-content 5px;

}

**GRID BEGINNER**

**HTML**

<!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" />

    <link rel="stylesheet" href="index.css" />

    <title>Document</title>

  </head>

  <body>

    <div class="container">

      <div class="item item--1">1: Orange</div>

      <div class="item item--2">2: Green</div>

      <div class="item item--3">3: Violet</div>

      <div class="item item--4">4: Pink</div>

      <div class="item item--5">5: Blue</div>

      <div class="item item--6">6: Brown</div>

    </div>

  </body>

</html>

**DEFAULT CSS**

.container {

  background-color: #eee;

  width: 100%;

  margin: 0 auto;

}

.item--1 {

  background-color: orange;

  color: #fff;

  grid-row-start: 2;

  grid-row-end: 3;

  grid-column-start: 2;

  grid-column-end: 3;

}

.item--2 {

  background-color: green;

  color: #fff; }

.item--3 {

  background-color: violet;

  color: #fff; }

.item--4 {

  background-color: palevioletred;

  color: #fff; }

.item--5 {

  background-color: royalblue;

  color: #fff; }

.item--6 {

  background-color: brown;

  color: #fff; }

**GRID CSS**

**Basic Position Grid Item**

in .container

    +Itemin hangi satirda yani rowda olmasini istiyorsak o satiri yaziyoruz 1. 2. satir gibi

grid-row-start: ;

    +Itemin hangi satirda yani rowda olmamasini istiyorsa orayi sinir olarak koyuyoruz. 2. 3. satir gibi.

grid-row-end: ;

    +Itemin hangi columnda olmasini istiyorsak orayi yaziyoruz.

grid-column-start: ;

    +Itemin hangi columda bitmesini yani olmamasini istiyorsak orayi yaziyoruz

grid-column-end: ;

grid-row: 2 / 3; (start - end)

grid-column: 4 / 5; (start - end)

(grid-row-start / grid-column-start / grid-row-end / grid-column-end)

grid-area; 1 / 3 / 2 / 4;

.container {

  display: grid;

+ Tum itemler ayri ayri tek bir satiri boydan boya kaplarlar

grid-template-columns: 1fr;

+ Satirin ilk elemanlari toplam fr sayisina yani 6 ya bolunur ve her fr kendi oraninda bir satiri boluserek paylasirlar.

grid-template-columns: 1fr 2fr 3fr;

+ 2 tane 200px genisliginde kutu yapar. Son kutuya atadigimiz 1fr satirda en fazla ne kadar genis alabiliyorsa o kadar yayilir.

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

  grid-template-rows: 150px 150px 150px;

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

  grid-template-rows: repeat(2, 200px) 200px;

  grid-template-columns: 150px 150px 150px;

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

  grid-template-columns: repeat(2, 200px) 200px;

+ Column ve Row aralarina bosluk birakmak

  grid-row-gap: 1rem;

  grid-column-gap: 1rem;

  grid-gap: 1rem;

}

**Spanning Grid Item**

.container {

+ Rowu 1. satir ve 2.satiri kaplayarak olusturur.

grid-row-start: 1;

grid-row-end: 3;

+ Column u 3. columndan yani siradan baslayarak 7. satira kadar kaplatir. (7 dahil degil)

grid-column-start: 3;

grid-column-end: 7;

+ Veya yukarida ki kisaltmalarida kullanabiliriz.

+Veya asagidaki gibi bir iteme verdigimizde 1. siradan yani columndan 3 kutu span yap. Kendi kutusundan sonra 2 kutu span yap yani. Eger ilk sayi 2. olursa kutu 2. siradan baslar ve devam eder. Column sayisi yetmezse alt satira gecmeden siniri asar ve devam eder.

grid-column: 1 / span 3;

+Veya 2. siradan baslatip daha sonra sira boyunca kac tane sira varsa hepsini kaplayarak devam ederek doldurarak gider asagidaki gibi kullanimda.

+Yani 2. siradan sonra saga column siniri kadar uzar.

+Yanni 3 column olsa grid-column: 2 / span 2; Bu kodla ayni anlama gelir asagidaki

grid-column: 2 / -1;

}

**Naming Grid Lines**

.container {

  display: grid;

    +Kisaca Grid Template Rows 'un anlami

        -header-start 'dan header-end 'e kadar 100px Row yuksekligi

        -header-end 'in bitis noktasi ile box-start 'in baslangic noktasi ayni yer

        -box-start 'dan box-end 'e kadar 200px Row yuksekligi

        -main-start 'dan main-end 'e kadar 400px Row yuksekligi

        -footer-start 'dan footer-end 'e kadar 100px Row yuksekligi

  grid-template-rows: [header-start] 100px [header-end box-start] 200px [box-end main-start] 400px [main-end footer-start] 100px [footer-end];

    +Kisaca Grid Template Columns 'un anlami

        -col-start ve col-end 'den 3 tane olustur ve col-start 'dan col-end 'e kadar Column genisligi 1fr olsun

        -repeat ile bu sekilde olusturulan col-start ve end ler olusturulduklari kadar cagrilirken belirtilmeleri icin kacinci deger olduklari sayi olarak yazilir. Ornegin col-start 1 veya col-end 1 veya col-start 3 veya col-end 2 gibi...

        -3 tane olusturdugumuz icin son col-end 3 'den grid-end 'e kadar 200px column genisligi olsun demisiz

  grid-template-columns: repeat(3, [col-start] 1fr [col-end]) 200px [grid-end];

}

**Kullanimlarida style 'da .container itemlari icinde bu sekilde**

header { grid-column: col-start 1 / grid-end; }

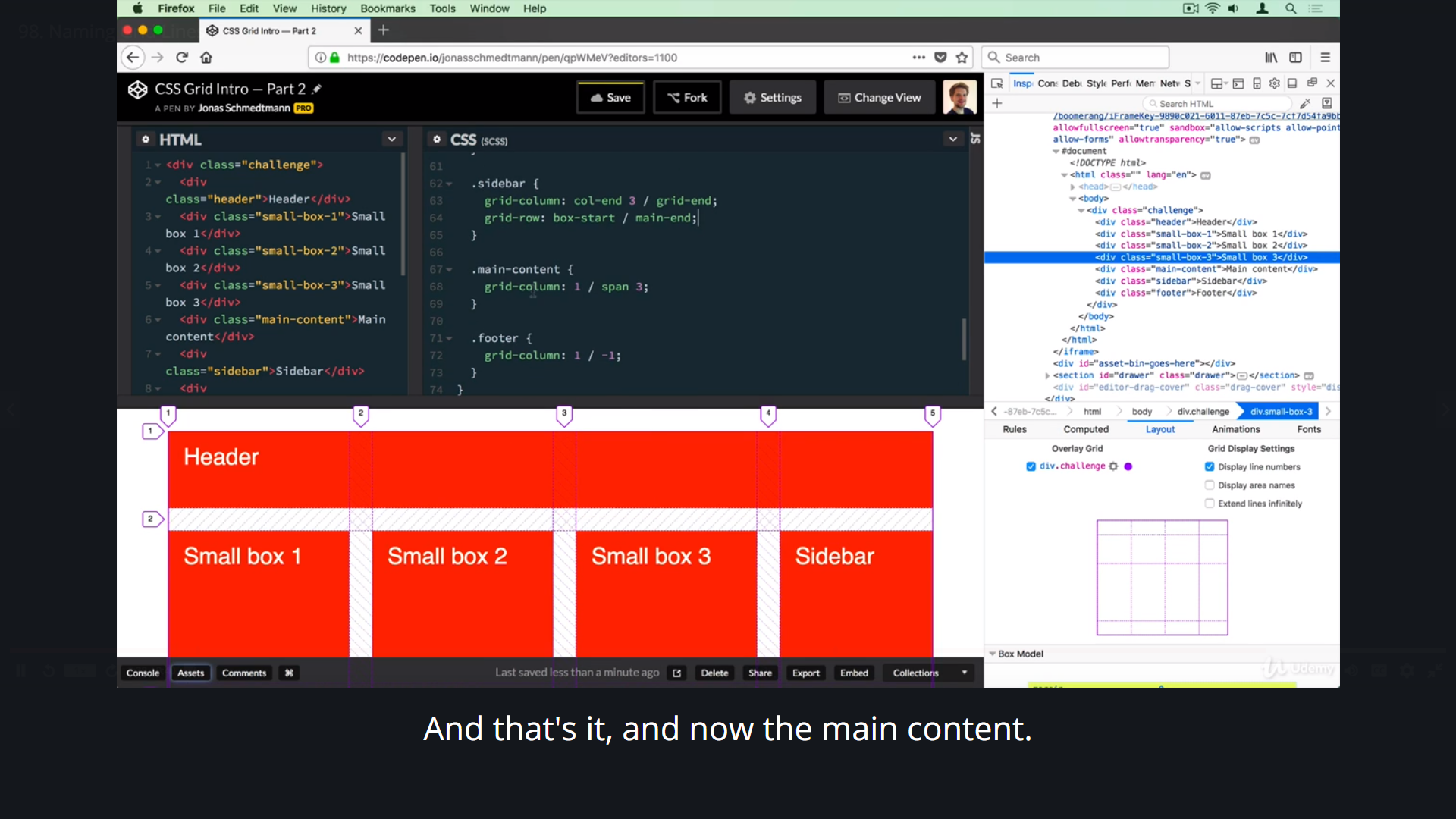
.sidebar {

  grid-column: col-end 3 / grid-end;

  grid-row: box-start / main-end;

}

.main-content { grid-column: col-start 1 / col-end 3; }



**Naming Grid Areas**

**Basit layout lar icin bu sistemi kullan**

.container {

  display: grid;

+grid-template-area da nokta konulan yerler bos birakilir.

  grid-template-areas:

    ". head head ."

    "box-1 box-2 box-3 side"

    "main main main side"

    "foot foot foot foot";

}

**Kullanimlarida style 'da .container itemlari icinde bu sekilde**

header { grid-area: head; }

.sidebar { grid-area: side; }

.small-box-1 { grid-area: box-1; }

.small-box-2 { grid-area: box-2; }

.small-box-3 { grid-area: box-3; }

.main-content { grid-area: main; }

footer { grid-area: foot; }

**EXCPILICIT GRID vs IMPLICIT GRID**

.container {

  display: grid;

+ Excpilicit Grid => Ozellikleri belirlenmis Row ve Column lardir

  grid-template-rows: 150px 150px;

  grid-template-columns: repeat(2, 1fr);

  grid-gap: 1rem;

+ Excpilicit Grid disinda kalanlara implicit Grid deriz.

+ Implicit Gridleri grid-auto-rows veya grid-auto-columns ile ozellikler verebiliriz.

  grid-auto-rows: 100px;

+ grid-auto-flow: row; iken grid-auto-columns: .5fr; deger verirsek implicit gridler

explicit gridlerin yarisi kadar genislik alir.

  grid-auto-columns: 0.5fr;

+ Gridleri Column olarak yada Row olarak islenmesini saglariz. Defaul olan Rowdur

  grid-auto-flow: column;

  grid-auto-flow: row;

+ row yanina dense yaparsak bazilarini tum satiri kapladigimiz icin tek kalan kucuk p[arcalari yan yana koyacakdir.

  grid-auto-flow: row dense;

}

**MINMAX / max-content – min-content – minmax(minvalue, maxvalue)**

.container {

  display: grid;

  grid-template-rows: repeat(2, 150px);

+ MaxContent bize diz icinde Mucahid yaziyorsa Mucahid yazisinin kapladigi alan kadar genislik almasini saglatir COLUMN 'un

+ MinContent ayni columnda bulunan en az contente sahip icerigin genisligini alir

  grid-template-columns: max-content 1fr 1fr min-content;

+ Budada ise 2 tane min contente sahip satir olusturur.

  grid-template-rows: repeat(2, min-content);

+ minmax ilede ilk once minimum ne kadar genislikte olacagini ve sonrada maximum ne kadar genislikte olacagini belirliyoruz

  grid-template-rows: repeat(2, minmax(150px, min-content));

}

**Alinging Grid Items To Grid Areas – Grid Itemslari Grid Alani Icinde Ortalama**

.container {

  display: grid;

+ Itemlar sadece css degerlerini koruyarak kapladiklari grid alanlarini dikeyde ortalarlar.

  /\* STRETCH / center / end / start \*/

  align-items: center;

+ align-items: stretch; ise Grid itemslarini default olarak geldigi degerdir

+ Tum grid items lar sadece icerikleri kadar gozukerek ve iceriklerinin tam ortasindan grid alanlarinin icerisinde dikey ve yatay olarak ortalanirlar.

     start => sol da ve dikey olarak ortada hizalanir

     end => sagda ve dikey olarak ortada

  justify-items: center;

}

.item--4 {

+ Align-items ve Justify-items in yaptiklarini tek bir elemana uygulamak istersek

  align-self: center;

  justify-self: center;

}

**Align Grid Tracks to Grid Container**

**Grid Sistemimizi Mevcut Alan Icinde Ortalama**

Yani Gridi uzatmadan genisletmeden Alanimizin genislemesine veya dikeylemesine ortasina veya yanlarina yerlestirmek

.container {

  width: 1000px;

  height: 1000px;

  display: grid;

  grid-gap: 1rem;

  grid-template-columns: repeat(2, 200px);

  grid-template-rows: repeat(2, 100px);

  /\* CENTER / RIGHT / LEFT / START / END / SPACE-BETWEEN / SPACE-ARROUND / SPACE-EVENLY \*/

  /\* Genislemesine Yatay olarak Horizontal ortalama \*/

  justify-content: space-evenly;

  /\* CENTER / RIGHT / LEFT / START / END / SPACE-BETWEEN / SPACE-ARROUND / SPACE-EVENLY \*/

  /\* Diker olarak Vertical Hizalama \*/

  align-content: space-evenly;

}

**Auto-fill and Auto-fit**

.container {

  display: grid;

  grid-template-rows: repeat(2, minmax(150px, min-content));

+ Auto-fill asagida 100px genisliginde columnlar olusturur olusturabildigi kadar. Genisligimiz 1000px oldugu icin her satira 10 tane columns kurar.

  grid-template-columns: repeat(auto-fill, 100px);

+ Auto-fit ise yine 10 columns olusturur yukaridaki gibi fakat auto-fillde columnlar yeterli item yokkende aktif isede auto-fit de ise bu columnlar yeni itemnlar gelinceye kadar gosterilmezler ve width 0 olarak saklanirlar

  grid-template-columns: repeat(auto-fit, 100px);

+ Fakat bu sekilde kullanimda ise oncelikli olarak max degeri olan 1fr i kullanacagi icin minmax olan icerikteki item sayisi 5 se 5 item tum satiri kaplar. 8ise 8 item tum satiri kaplar. Kalan 5 veya kalan 2 item geldikce yanlarina ayni satirda eklenirler.

  grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));

}