**SASS ( Syntactically Awesome StyleSheets )**

Sass 2 uzantili sekilde kullanilabiliyor. .scss .sass.

.scss daha sik tercih ediliyor

Sass tarayicilar direk okuyamiyor onun icin node-sass kurmamiz sart

**SASS in Bize Sagladiklari**

\*degiskenler tanimlayabilir ve onlari cagirip kullanabiliyoruz

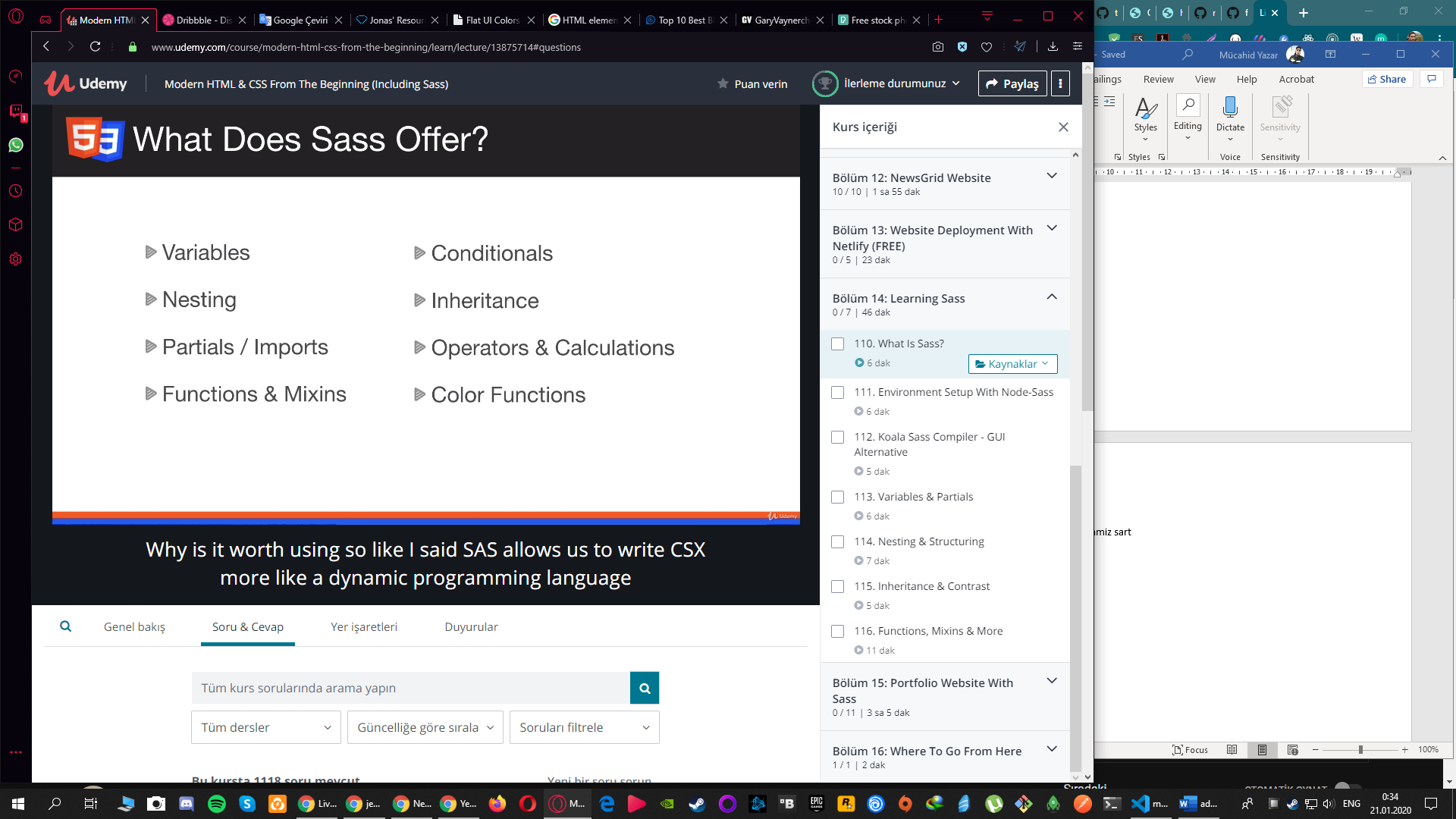
\*baska css dosyalarini iceri import edip kullanabiliyoruz

\*fonksiyon seklinde kullanimlar yapabiliyoruz

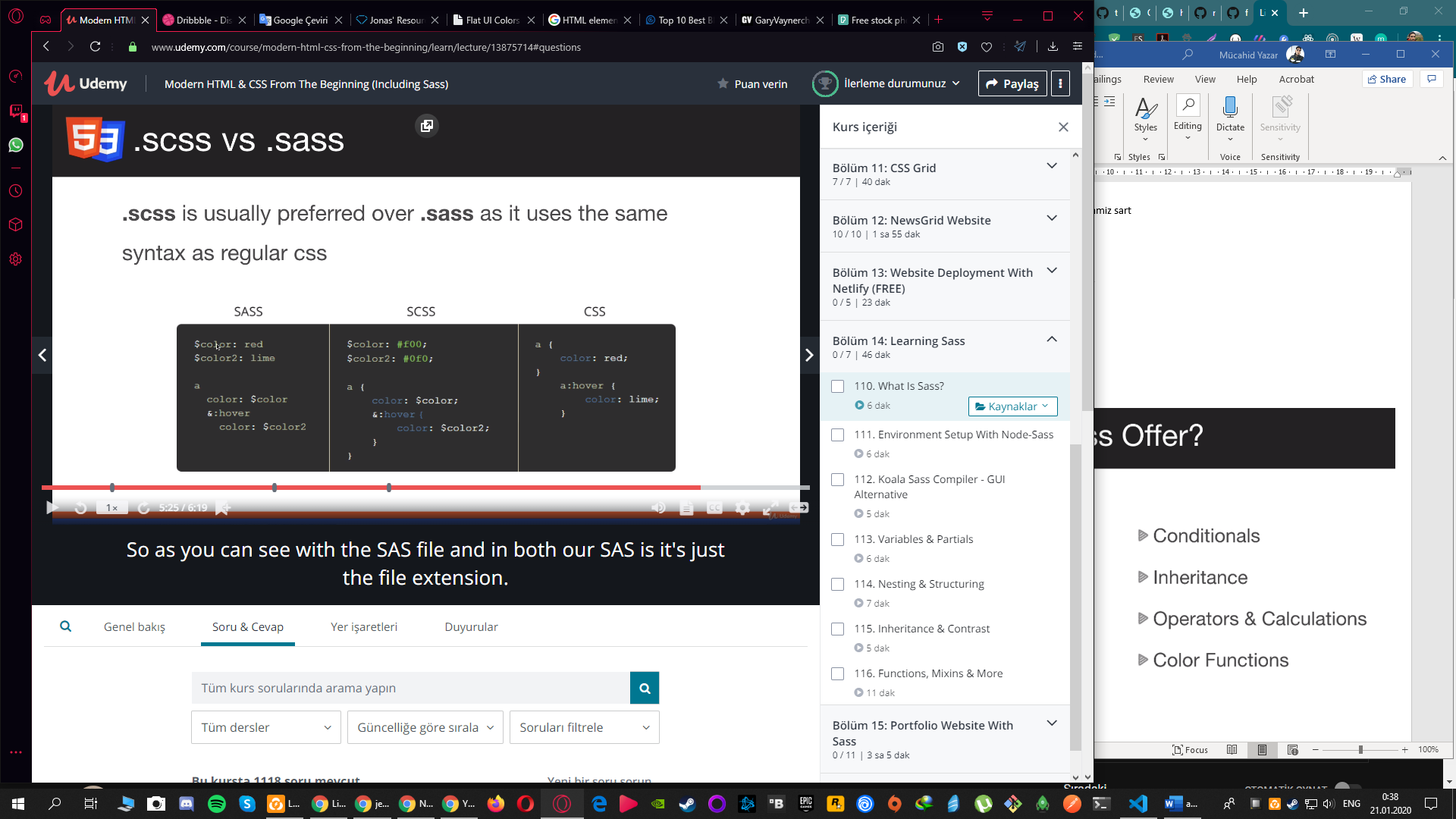
\*if else kosullari koyabiliyoruz

\*Kendi css bloklarimizi olusturabiliyoruz

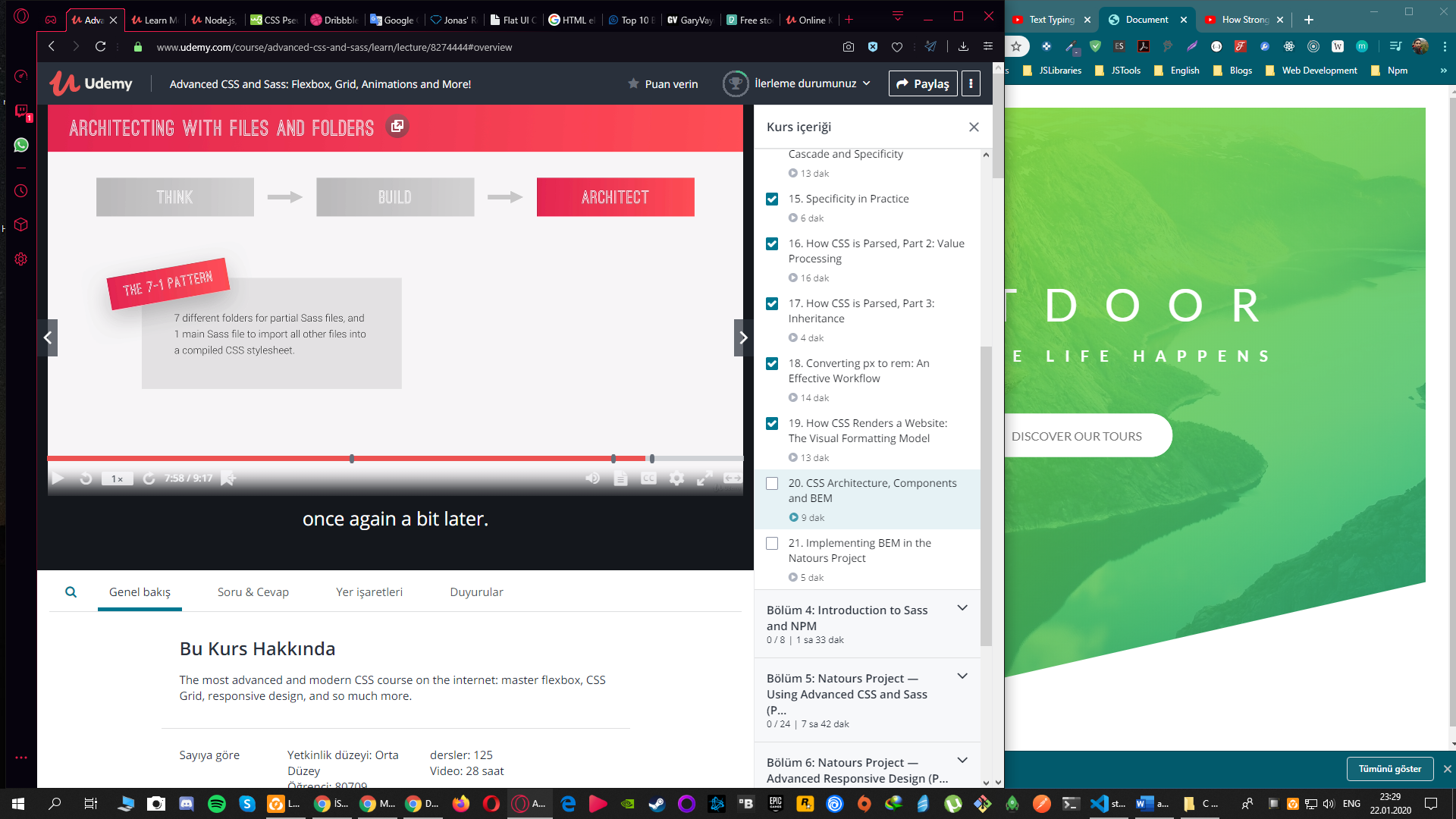
\*renk fonksiyonlari tanimlayabiliyoruz

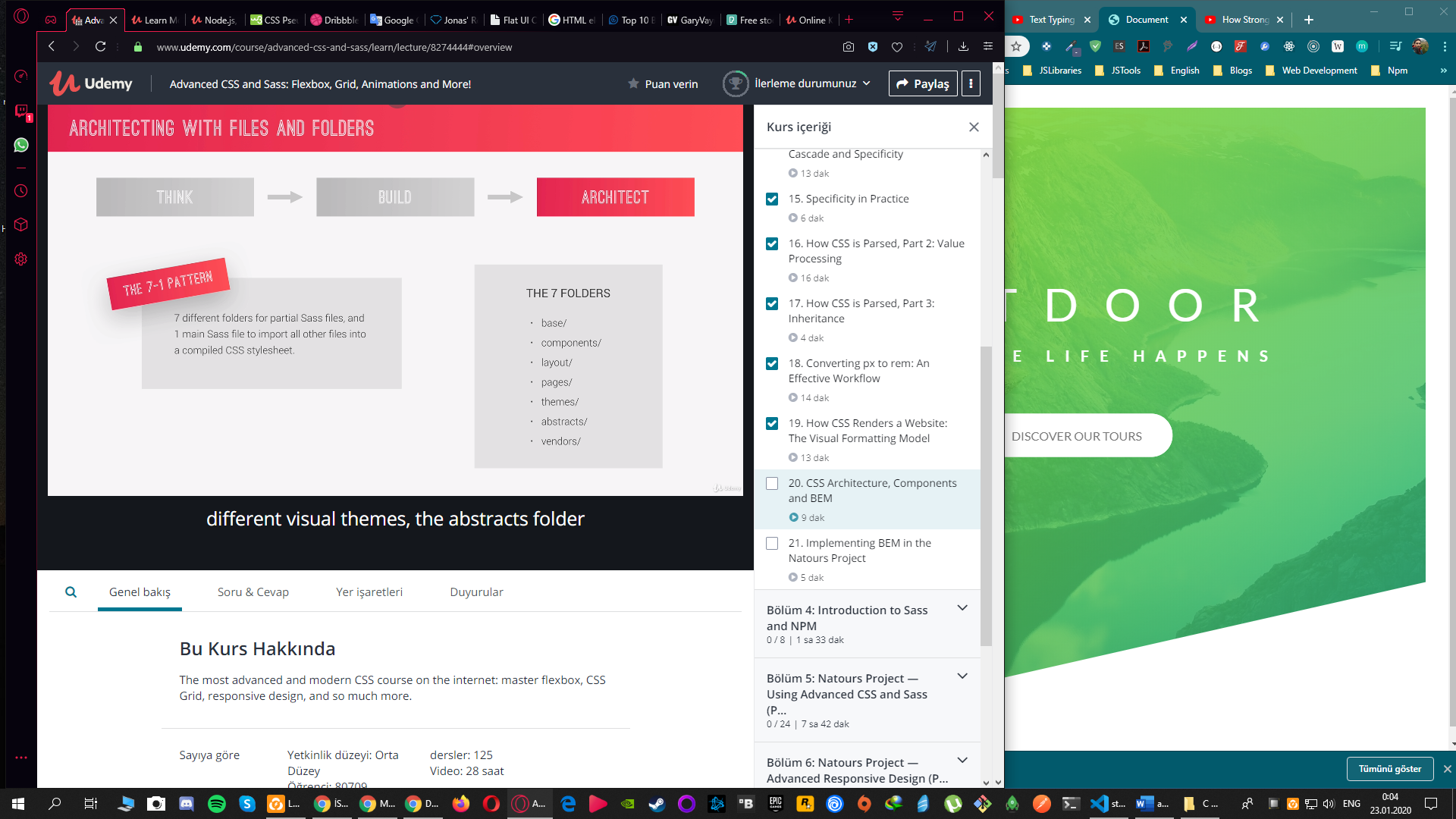


**SASS vs SCSS vs CSS**



SASS KULLANIMI





SASS Kurulum

1. Npm init y
2. Npm i node-sass
3. Package.json => scripts => “sass”: “node-sass -w scss/ -o dist/css/ --recursive”

Yukarida scss/ ile scss dosyalarimizin bulundugu konumu belirliyoruz

Dist/css/ ilede onlari css donusturdukten sonra birakacagi yeri belirliyoruz. Biz burada ki css i html sayfamiza ekleyecegiz

-w watch demek

--recursive de sanirim otomatik olarak her degisikliktenb sonra bize gerek kalmadan kendisi degisiklikleri css e gonderiyor

Ve npm run sass ile baslatiyoruz.

SASS OZELLIKLERI

* SASS ici degisken tanimlayarak kullanim

$color: blue;

$priamry-color: steelblue;

$secondary-color: skyblue;

$light-color: #f4f4f4;

$dark-color: #333;

$font-stack: Arial, Helvetica, sans-serif;

body {

    background: $light-color;

    color: $dark-color;

    font-family: $font-stack;

}

* Yada \_variables diye bir degisken olusturup,

$ ile tanimlanan degiskenleri oraya alip,

Daha sonra @import variable yazip

Degiskenlerimizi baska dosyada tutup import ederekde calsiabiliriz.

* Eger \_ yi variables a koymazsak node-sass bunu css klasorunun icine variable.css olarak compiled ederdi yani cikartirdi.
* Nesting yapisi ile de ic ice kodlar yazabiliriz asagidaki gibi
* Burada header icinde h1 i aliyoruz yani seciyoruz sadece.

header {

    background: $dark-color;

    color: $light-color;

    padding: 1rem;

    h1 { //Bu bır NESTED SELECTORS dur

        text-align: center;

    }

}

**& ile EKLEME YAPMA**

* Asagida ise .section a padding ve h3unu bazi degisiklikler yapmisiz. Onun altinda ise &-a yaparak .section-a yi seciyoruz. Cunku & sonra yazilan seyler &’nin scss icindeki parent elementine eklenir yani alt ornekte .section’a eklenir. Ve secim yapilir.

.section {

    padding: 3rem;

    h3 {

        font-size: 2rem;

    }

    &-a { // => .section-a

        background: $primary-color;

        color: #fff;

    }

}

* Yada bu sekilde ekleme yapariz ve hover i seceriz

a {

    color: #333;

    &:hover {

        color: coral;

    }

}

**% ile TOPLU OZELLIK TANIMLAMA @EXTEND ILE IMPORT ETMEK**

* Yada asagidaki gibi basina % isareti koyarak %btn-shared ile bir degisken tanimlariz ve daha sonra bu degiskenin ayni ozelliklere sahip olmasini istediklerimize @extend %btn-shared yaparak ozelliklerini aktaririz. Asagidaki gibi

%btn-shared {

    display: inline-block;

    padding: 0.7rem 2rem;

    border: none;

    cursor: pointer;

    text-decoration: none;

    margin-top: 1rem;

}

.btn {

    &-light {

        @extend %btn-shared;

        background: $light-color;

        color: #333;

    }

    &-dark {

        @extend %btn-shared;

        background: $dark-color;

        color: #fff;

    }

}

**BAGLANTILI ILISKILI ELEMENT LERDA EXTENDS, BAGLANTILI ILISKILI OLMAYAN ELEMENT LERDE MIXIN KULLAN**

**FUNCTIONS**

* \_functions.scss adinda bir dosya olusturalim.

Ve icine main.scss de kullanmak icin asagidaki fonksiyonu kuralim

* @fonksiyon tanimlamamiza yardimci olur.
* Lightness bir javascript komutudur aslinda rengin tonunun ne kadar acik oldugunu belirliyor ve 0ile100 arasinda bir deger donuyor bize.
* Yani burada demisiz ki set-text-color adinda bir fonksiyon olusturmusuz. Ve bunun icinde gelen degeri kontrol et demisiz. Eger gelen renk 50 den daha buyukse KOYU bir renk donecek, daha kucukse acik bir renk donecek.

// Set Text Color

@function set-text-color($color) {

    @if(lightness($color) > 50) {

        @return #000;

    } @else {

        @return #fff;

    }

}

* Kullanma isinide asagidaki gibi yapiyoruz. Kirmizi ile yazdim.

header {

    background: $dark-color;

    color: set-text-color($dark-color);

    padding: 1rem;

    h1 {

        text-align: center;

    }

}

**MIXIN**

--webkit-transform: rotate(-20deg);

-ms-transform: rotate(-20deg);

transform: rotate(-20deg);

* Transform tum tarayicilarda olmadigi icin yukaridaki kodlarin hepsini tanimlamamiz gerekiyor kullandigimiz zaman diger tarayicialrdada sorunsuz calismasi icin. Fakat bunun yerine mixini kullanarak onlari baska yerde tanimlar ve main.scss dede kullanabiliriz.

1. Once \_function.scss i olusturup @import ‘functions’ diyerek import ediyoruz main.scss dosyamiza

// Transform mixin

@mixin transform($property) {

    --webkit-transform: $property;

    -ms-transform: $property;

    transform: $property;

}

1. Sonra yukaridaki sekilde mixini cagirip transform adinda degiskenimizin ismini tanimliyoruz ve parantez icinde gelecek tanimlamalarimizda kullanacagimiz property i tutuyoruz. Bu property ‘i main.scss den biz yollayacagiz.
2. Daha sorna main.scss e gelip, ornegin hover olunca mixini kullanalim diyerek asagidaki gibi @include ile mixini cagiriyorum. @include dedikten sonra tanimladigimiz mixin adini yaziyoruz oda transform. Ve bu degiskene gonderecegimiz property i : koyduktan sonra rotate(20deg) gibi asagidaki gibi yaziyoruz.

&:hover {

            @include transform(rotate(20deg));

            // aslinda yukarida olan asagidakinin aynisi

            // transform: rotate(20deg);

        }

**DARKEN and LIGHTEN**

Bu sekildede fonksiyonlu koyulastirma vveya renk acma islemleri yapabiliriz. Amountta yazan miktar % miktaridir.

            background-color: darken($color: $light-color, $amount: 10);

            background-color: lighten($color: $dark-color, $amount: 5);

backface-visibility: hidden;

//Animasyon bittikten sonra konum bozulmasini duzeltiyor

Tiklama efekti

Hover oldugumuzda 3px yukari gider buton

Tikladigimizda ise oldugu yerden 1 px asagi iner

ve tiklama efekti olusur

&:hover {

        transform: translateY(-3px);

    }

    &:active { //Tiklandiginda

        transform: translateY(-1px);

    }

Bununlada disariya acilan ayri bir beyaz golge veririz en son acildiginda opacity 0 olur hoverafter sayesinde

    &::after {

        content: '';

        display: inline-block;

        height: 100%;

        width: 100%;

        border-radius: 100px;

        position: absolute;

        top: 0;

        left: 0;

        z-index: -1;

        transition: all .4s;

    }

    &:hover::after {

        transform: scaleX(1.4) scaleY(1.8);

        opacity: 0;

    }

**@SUPPORTS KULLANIMI**

Bu ozellik sayesinde desteklenen ozelligi destekleyen taraticilara kullandirirken desteklemeyenlere default olarak degerler kullandirabiliriz.

.heading-secondary {

  font-size: 3.5rem;

  text-transform: uppercase;

  font-weight: 700;

  display: inline-block;

  letter-spacing: 0.2rem;

  transition: all 0.2s;

//Burada supportsdan sonra yazilan cok onemli degil sadece destekleyip desteklemedigi kosulunu ogrenmek icin bunu yaziyoruz. Rastgele degerler verilebilir.

  @supports (background-clip: text) or (-webkit-background-clip: text) {

    color: transparent;

    background-image: linear-gradient(

      to right,

      $color-primary-light,

      $color-primary-dark

    );

    background-clip: text;

    -webkit-background-clip: text;

  }

Mesela yukarida background-clip veya -webkit-background-clip desteklemiyorsa heading altindakileri kullan demisiz destekliyorsada support icindekiler uygulansin demisiz.

**ORNEK SUPPORT KULLANIMLARI**

  @supports (-webkit-backdrop-filter: blur(10px)) or

    (backdrop-filter: blur(10px)) {

    -webkit-backdrop-filter: blur(10px);

    backdrop-filter: blur(10px);

    background-color: rgba($color-black, 0.3);

  }

    @supports (clip-path: polygon(0 0)) or (-webkit-clip-path: (0 0)) {

      //Kenarlarini keser ve yaninda yaZi varsa bu kesilmis alanlara kaydirilir. Yani burada daire seklinde bir kesme islemi yaptik

      -webkit-shape-outside: circle(50% at 50% 50%);

      shape-outside: circle(50% at 50% 50%);

      //Burada ise kesme islemi sadece goruntu olarak olur. shap-outside da kesme islemi fiziki olarak olur ama goruntude hala gozukmeye devam eder. Bu sekilde yine keserek goruntu olarakda kesilen yerlerin transparent gibi gozukmemesini saglariz

      -webkit-clip-path: circle(50% at 50% 50%);

      clip-path: circle(50% at 50% 50%);

      border-radius: none;

    }

**SETTING UP CSS FILE**

Tum css dosyalarini birlestirmek toplamak ve derlemek istiyorum bunun icin asagidaki adimlari takip ediyoruz

  "scripts": {

    "devserver": "nodemon app.js",

    "watch:sass": "node-sass -w public/scss/main.scss -o public/css/main.css --recursive",

//Bu sekildede iki komutu npm-run-all sayesinde calistirabiliriz. –parallel ise ayni anda calismalarini saglar. Eger parallel olmasaydi asagidaki gibi sirayla calisirdi.

"start": "npm-run-all --parallel devserver watch:sass",

//Once scss dosyamizin ciktisini aliyoruz

    "compile:sass": "node-sass public/scss/main.scss -o public/css/style.comp.css",

//Daha sonra concat adli npm modulunu yukluyoruz. Bununla css dosyalarimizi birlestirecegiz.

//Ilk dizin cikti yapilacak dizin

//Daha sonra birlestirecegimiz main.csss in ciktisi main.css ile icon larin oldugu icon-font.css I secerek birlestiriyoruz.

    "concat:css": "concat -o public/css/style.concat.css public/css/icon-font.css public/css/style.comp.css"

//Burada ise Prefix islemini yapacagiz. Prefix bizim icin -webkit gibi uyumluluk destek eklerini otomatik olarak ekler

//Prefix postcss modulunun bir eklentisidir. Bu yuzden postcss-cli ve autoprefixer eklentisini yukluyoruz

//Daha sonra asagidaki gibi script olusturuyoruz.

//Anlami postcss I calistir / autoprefixer I use ile kullan / -b Browserlari sec ‘last 10 versions’ yani son 10 eski tum browser versiyonlarini sec / prefix yapilacak css I sec / -o yaz ve output konumunu yaz

"prefix:css": "postcss --use autoprefixer -b \"last 10 versions\" public/css/style.concat.css -o public/css/style.prefix.css"

Ve son adim olarak prefix yapilmis css dosyamizin node-sass ile compress ediyoruz yani sikistiriyoruz. Boylelikle cokdaha kucuk bir boyut alacak ve tarayicimiza hiz kazandiracak.

Once node-sass I cagiriyoruz. Sonra prefix edilmis css dosyasini seciyoruz ve daha sonra output noktasini ve output ismini ayarliyoruz. Ve outputstyle ini compressed olarak ayarliyoruz.

"compress:css": "node-sass public/css/style.prefix.css public/css/style.css --output-style compressed"

//Tum bu yukaridaki adimlari tek tek de sirasiyla calistirip production ciktisini alabiliriz fakat bunun yerine npm-run-all modulunu yukluyoruz. Bunu baska seylerdede kullanabilirsiniz. Live servber ile node sass I birlikte calistirma gibi 😊

//Yukledikten build script I olusturuyoruz ve icine npm-run-all sonrasindada calistirilacak scriptleri sirasiyla yaziyoruz.

"build": "npm-run-all compile:sass concat:css prefix:css compres:css"

  },

**SCSS and MEDIA QUERIES**

**@CONTENT and INCLUDE**

Burada @content => responde-phone nin include edildigi yerde icine yazilan kodlardir.

@mixin respond-phone {

  @media (max-width: 600px) {

    @content;

  }

}

html {

  // This defines what 1 rem is

  font-size: 62.5%;

  @include respond-phone {

    font-size: 50%;

  }

}

**SCSS MEDIA QUERY SYSTEM**

Aslinda normal css deki gibi de yapabiliriz fakat bu en guzel ve en pratik sistem

Asagida mixin ile respond MIXIN’i olusturuyoruz. Icinde breakpoint gelecek diyoruz. Ve breakpoint tab-land e esitse diyoruz @media (max-width) css ‘i icine @content ten gelen degeri uygula diyoruz

//Burada em kullanacagiz cunku kullanicilar tarayici ayarlarini degistirdiklerinde px degerleri degisebiliyor.

//Rem kullanmamamizin sebebi ise rem bazi tarayicilarda failed olabiliyor.

//1EM 'i 16px alarak hesaplama yapabilirsiniz.

@mixin respond($breakpoint) {

  @if $breakpoint == phone {

    @media (max-width: 37.5em) { @content; } //600px

}

  @if $breakpoint == tab-port {

    @media (max-width: 56.25em) { @content; } //900px

}

  @if $breakpoint == tab-land {

    @media (max-width: 75em) { @content; } //1200px

}

  @if $breakpoint == big-desktop {

    @media (min-width: 112.5em) { @content; } //1800px

  }

}

Buradada respond(tab-land) olarak font-size 56.25% olarak gonderiyoruz

html {

  // This defines what 1 rem is

  font-size: 62.5%; //I want 1 rem => 10px, 10/16 => 62.5%

  //width < 1200px

  @include respond(tab-land) {

    font-size: 56.25%; //I want 1 rem => 9px, 9/16 => 56.25%

  }

  //width < 900px

  @include respond(tab-port) {

    font-size: 50%; //I want 1 rem => 8px, 8/16 => 50%

  }

  //width > 1800px

  @include respond(big-desktop) {

    font-size: 75%; //I want 1 rem => 12px, 12/16 => 75%

  }

}

Diyelim ki sayfa genisligimiz 700px hangi font-size uygulanir sizce? Tab-port’da tab-land de bunun icin uygun. Ikiside uygulanabilir. Iste burada alt satirda yazdigimiz daha oncelikli kabul edilir ve o uygulanir. Alt satirada daha yakin deger olan tab-port uygulanir. O yuzden siralama bu sekilde olur.

**CALC KULLANIMI and GRID SYSTEM**

.row {

  max-width: $grid-width;

  background-color: #eee;

  margin: 0 auto;

  &:not(:last-child) {

    margin-bottom: $gutter-vertical;

  }

  @include clearfix;

Mixinde olan hali altta

@mixin clearfix {

  &::after {

    content: "";

    display: table;

    clear: both;

  }

}

  [class^="col-"] {

    background-color: orangered;

    float: left;

    &:not(:last-child) {

      margin-right: $gutter-horizontal;

    }

  }

  .col-1-of-2 {

    width: calc((100% - #{$gutter-horizontal}) / 2);

  }

  .col-1-of-3 {

    width: calc((100% - 2 \* #{$gutter-horizontal}) / 3);

  }

  .col-1-of-4 {

    width: calc((100% - 3 \* #{$gutter-horizontal}) / 4);

  }

  .col-2-of-3 {

    width: calc(

      2 \* ((100% - 2 \* #{$gutter-horizontal}) / 3) + #{$gutter-horizontal}

    );

  }

  .col-2-of-4 {

    width: calc(

      (100% - 3 \* #{$gutter-horizontal}) / 4 \* 2 + #{$gutter-horizontal}

    );

  }

  .col-3-of-4 {

    width: calc(

      (100% - 3 \* #{$gutter-horizontal}) / 4 \* 3 + 2 \* #{$gutter-horizontal}

    );

  }

}

**SELECTOR KULLANIMI**

^ isareti ile demek istedigimiz col- ile baslayan tum classlari sec

  [class^="col-"] {

    background-color: orangered;

    float: left;

    &:not(:last-child) { //Son child haric hepsine asagidaki margin right i ekle

      margin-right: $gutter-horizontal;

    }

  }

**CSS FILTER KULLANIMI**

  &:hover &\_\_img {

    transform: translateX(-4rem) scale(1);

    filter: blur(3px) brightness(80%);

  }

**HANGI RENKTEN NE KADAR YUZDE KULLANILSIN**

.book {

  background: linear-gradient(

//Burada da 105 derecelik aciyla renk dagilsin demisiz

      105deg,

//color-white in opacity 0.9 olsun. %0 oraninda kullanilsin. Altindaki de %50 oraninda kullanilsin.

      rgba($color-white, 0.9) 0%,

      rgba($color-white, 0.9) 50%,

      orangered 50%

    ),

    url(/public/img/nat-10.jpg) no-repeat center center / cover;

  border-radius: 3px;

  box-shadow: 0 1.5rem 4rem rgba($color-black, 0.2);

  height: 50rem;

}