2022

Web Programlama Kulübü

16.12.2022

CSS Nedir ve Nasıl Kullanılır?



# CSS

CSS'de arka plan renklerini değiştirmekten ve nesnelerin neden olması gerektiği gibi hizalanmadığını merak etmekten daha fazlası vardır. Ancak JavaScript framework'leri ve kütüphaneleri gibi diğer araçlara zaman ayırdığımız için, CSS genellikle gözden kaçan ve arka planda kalan **görsel bir iletişim** aracıdır. Genellikle temel çalışma yeterliliğine sahip CSS'i alırız ve acil hedeflerimize ulaşır ulaşmaz daha derine inmeden bırakırız. Sonuç olarak uygulama genelinde yapı, mimari ve uyum gibi şeyleri göz ardı etme eğilimindeyiz. Biz bu yazımızda CSS hakkında bilmemiz gerekenleri ele alacağız ve şimdi en baştan başlayalım.

# CSS (Cascading Style Sheets) Nedir?

CSS (Basamaklanmış Stil Katmanları), **web sayfasını şekillendirmek için kullanılan bir kodlama türüdür**. CSS kullanarak **web sayfasının görünümünü ve düzenini değiştirebiliriz**. Bir web sitesinin görünümünün masaüstü bilgisayarlar, tabletler ve mobil cihazlar gibi **farklı ekranlarda nasıl değiştiğini de tanımlayabiliriz**. CSS, *C++ veya JavaScript* gibi bir **programlama dili** **değildir**. Bunun nedeni, CSS'in amacının web tarayıcıları için biçimlendirmeye(HTML'e) stil vermek olmasıdır. HTML yalnızca içeriği "işaretleyen" bir dildir yani, belgeye belirli bir görsel ve yapısal biçimlendirme verir.

Hem HTML hem de CSS doğaları gereği bildirimseldir. Bu, yalnızca bir şeyleri açıkladığı ve herhangi bir işlem yapmadığı anlamına gelir. Tarayıcılar CSS'i görüntüleme için yorumlarken, bunu doğrusal bir şekilde yapar. Bunun nedeni, CSS'in dosyayı yukarıdan aşağıya okuyan tarayıcıya talimatlar vermesidir. Bir kural belirlendiğinde ve daha sonra dizin aşağıya doğru değiştirildiğinde, tarayıcı ekranda görüntülenecek nihai sonuç olarak en son değerlendirmeyi alır. CSS'deki C'nin ifade ettiği şey budur. CSS'in cascade(kaskad)etkisi, tarayıcıyı, kuralın fiziksel olarak ne zaman gerçekleştiğine bağlı olarak, kuralları art arda ve geçersiz kılacak şekilde görüntüleyecek şekilde ayarlar. Cascade(kaskad), öğe için birden çok CSS bildirimi ve birden çok stil sayfasının bir kombinasyonu olduğunda, belirli bir öğe için uygulanan stiller arasındaki çakışmayı çözme anlamına gelir. Sonuç olarak, CSS göründüğü kadar kolay değildir. Anlamadan kullanmaya çalışırsanız web geliştirmede zorluk yaşarsınız. Bu nedenle, CSS öğrenmek bir programlama dili öğrenmek kadar zor olmasa da yine de önemlidir.

# CSS Syntax

# 

İlk CSS kodumuzu yazalım. Örneğin, h1 etiketinin rengini değiştirmek istiyorum.

<h1> I'm a Header</h1>



Öncelikle, CSS'e HTML öğesini nasıl bulacağını söylememiz gerekir. Bunu selector(seçici) denen bir özellik ile yapabiliriz. CSS'de bir **selector, HTML öğelerini etiket adı, sınıf adı, kimliği ve çok daha fazlasına göre bulmak için kullanılır**.

O zaman şimdi HTML öğesi için bir selector tanımlayalım ve ardından selector parantezleri arasında her biri noktalı virgülle biten CSS kurallarını bildirelim.

h1 { color: red; }



Böylelikle, tanımlı selector'e (h1) bağlı olarak, CSS artık yeni kuralların nereye uygulanacağını anlayabilir.

Bunların yanı sıra CSS kodlarımızı .css uzantılı bir dosya içerisinde tutar ve bu dosyayı HTML sayfası içerisine basit bir kodla çağırabiliriz. Örneğin CSS dosyamızın adı **"hello.css"** olsun ve bu dosyayı Html sayfaya eklemek için HTML içine yazmamız gereken kod şu şekildedir.

<head> <link rel=”stylesheet” type=”text/css” href=”hello.css” /> </head>

Bir CSS selector tanımlamanın birçok farklı yolu vardır, aşağıda selector türlerinin bazı örneklerini görebilirsiniz:

* **Class Selector:** HTML öğelerini sınıf özelliklerine göre bulur.
* **ID Selector:** Öğeleri belirli kimliğine göre bulur.
* **Element Selector:** Öğeleri etiket adlarına göre bulur.

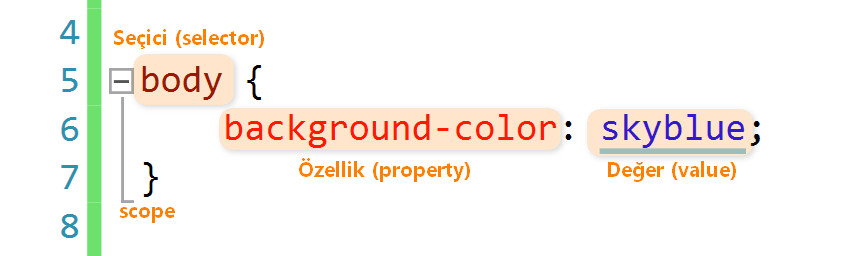
CSS teknik olarak kolaydır. Anlaşılması gereken sınırlı sayıda kural vardır. Bununla birlikte, CSS'i zorlaştıran potansiyel kural kombinasyonudur. Görsel gereksinimler daha karmaşık hale geldikçe, CSS kurallarınızın sürdürülebilirliğini kontrol altında tutmak için yaratıcı ancak yapılandırılmış çözümlere ihtiyaç vardır.

Tasarım yönleri pazarlamaya ve UX tasarımcısına bırakılabilirken, front-end geliştiricilerin tasarımın nasıl çalıştığını anlamaları da iyidir. Tasarımcıların tümü geliştirici değildir ve bazen tarayıcıların ve piksel rehberliğinin daha ince gereksinimlerini anlamayanlarla karşılaşıyoruz. Dijital tasarımcıların stil sayfalarının inceliklerini ve potansiyel karmaşıklığını öğrenmesi gerektiği kadar, tasarım dilinde de konuşabilmelisiniz.

# HTML Sayfamıza CSS Dahil Etme

CSS, web sitenizin ekranda nasıl görüneceğini belirleyen belirli stil kurallarına sahip dosyalardır. CSS kuralları, web sitenizin HTML dosyalarına **3 farklı şekilde** uygulanabilir. **External yani harici bir stil sayfası, Internal yani dahili bir stil sayfası veya Inline yani satır içi stil kullanabilirsiniz.** Her yöntemin belirli kullanımlara uygun avantajları vardır.

**External stil sayfası, bir web sayfasından bağlanan bağımsız bir .css dosyasıdır**. External stil sayfasının avantajı, bir kez oluşturulabilmesi ve birden çok web sayfasına uygulanabilmesidir. Site tasarımınızda geniş çaplı değişiklikler yapmanız gerekirse eğer, stil sayfasında tek bir değişiklik yapabilirsiniz ve bu değişiklik tüm bağlantılı sayfalara uygulanarak zamandan ve emekten tasarruf sağlar.



**Internal stil sayfası, HTML dosyasının <head> bölümünde sayfa için CSS kural kodlarını barındırır**. Kural kodları yalnızca o sayfa için geçerlidir, ancak sayfa kodunda birden çok öğeye stil uygulamak için kullanılabilecek class ve id’leri yapılandırabilirsiniz. Yine, CSS kodunda yapılacak tek bir değişiklik, sayfadaki tüm etiketlenmiş öğelere uygulanacaktır.

**Inline stili, belirli bir sayfa öğesine stil vermek için, belirli bir HTML etiketi içinde kullanılır**. Hızlı ve kalıcı değişiklikler için kullanışlıdırlar, ancak bir tasarım değişikliği yapmaya karar vermeniz durumunda, oluşturduğunuz her Inline stilin ayrı ayrı düzenlenmesi gerektiğinden, External ve Internal stil sayfalarından daha kullanışlı değildir ve diğerlerine göre çok fazla tercih edilmez.

# 1-) External CSS Kullanımı

Yukarıda bahsettiğimiz gibi External CSS, normal HTML dosyanızdan bağımsız bir dosyadır ve .css uzantılıdır. CSS dosyanızı oluşturduktan sonra mutlaka HTML dosyanıza linklemeniz gerekir. Yoksa CSS dosyanız işlevsiz kalacak ve HTML sayfanızda yaptığınız değişiklikleri göremeyeceksiniz. Hadi gelin bir de örnek üzerinde görelim.

.css uzantılı dosyamızı oluşturduktan sonra içine kodlarımızı bu şekilde yazıyoruz:

body {

background-color: #f7cac9;

}

.firstDiv, .secondDiv, .thirdDiv {

border: 7px outset gold;

background-color: lightblue;

text-align: center;

}

Daha sonra HTML sayfamıza geri dönüyoruz ve yazdığımız bu External CSS kodunu <head> bölümüne aşağıdaki gibi linkliyoruz. İşte bu kadar.

<link rel="stylesheet" href="style.css">

# 2-) Internal CSS Kullanımı

Internal CSS, stil kodlarının direk HTML dosyasının içinde <head> bölümünde kullanıldığı bir yöntemdir. <head> bölümünün içinde bir <style> etiketi açtıktan sonra içine gerekli CSS kodlarını yazabilirsiniz.

**Örnek vermek gerekirse; kodunuz şu şekilde görünecektir:**

<!DOCTYPE html>

<html>

<head>

<style>

body {

background-color: #f7cac9;

}

.firstDiv {

border: 7px outset gold;

background-color: lightblue;

text-align: center;

}

</style>

</head>

<body>

<div class="firstDiv">

<h2>Lorem ipsum</h2>

<p>Lorem ipsum dolor sit amet consectetuer</p>

</div>

</body>

</html>

# 3-) Inline CSS Kullanımı

Inline yani satır içi stiller, doğrudan HTML kodunuzdaki herhangi bir öğeye uygulanır. Stil niteliği ve ardından normal CSS özellikleri bir HTML etiketi içinde belirtilir. Kullanımı aşağıdaki örnekte gösterildiği gibidir:

<!DOCTYPE html>

<html>

<head>

</head>

<body>

<div style="border-width:7px; border-style:outset; border-color:gold; background-color:lightblue; text-align:center;">

<h2>Lorem ipsum</h2>

<p>Lorem ipsum dolor sit amet consectetuer</p>

</div>

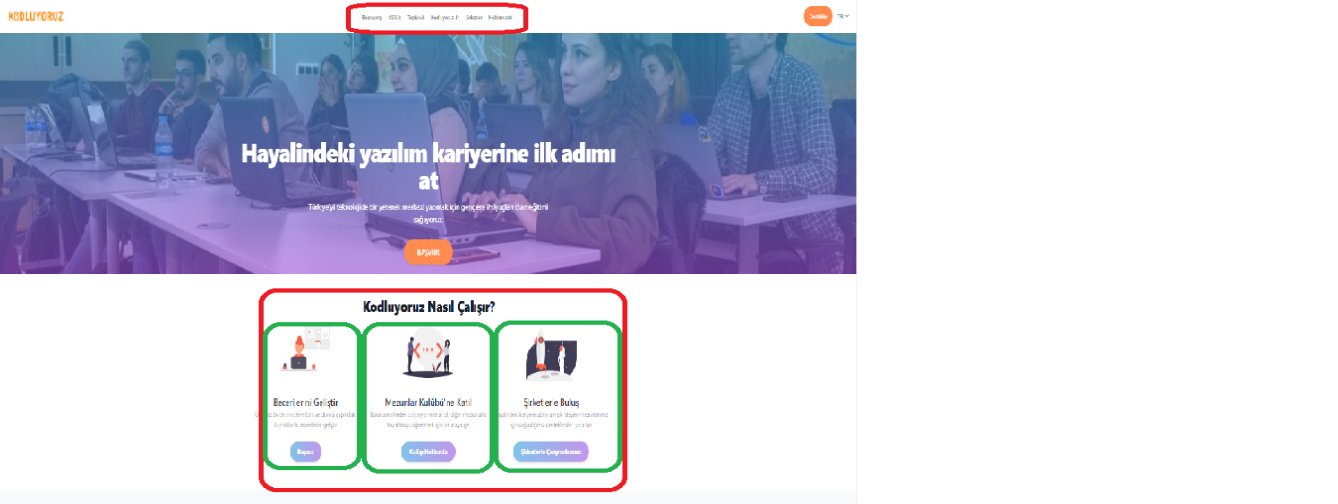
</body>

</html>

# CSS Kutu Özellikleri(Margin, Padding, Width, Height) Kullanımı

CSS kutu özelliklerini iyi kavrayabilmek ve kullanabilmek için "kutu model" konusunu incelemek gerekir. Gelin önce "kutu modeli" konusuna bakalım.

# CSS Kutu Modeli



Resimde Kodluyoruz’un anasayfasını görüyoruz. İşaretlediğimiz alanlar birer HTML elementleri ve bir yapıyı oluşturan lego parçalarından sadece birkaçı. Kutu denmesinin sebebi de bir bütünü oluşturan lego parçaları veya kutu gibi olması. Bu kutuların aralarında mesafelerin ve içeriklerinin de kendi aralarında bir düzen olduğunu görmekteyiz. Bunun sebebi "kutu özelliklerinin" kullanılıyor olması. Özellikle yeşil kutuların içindeki yazıların belli bir mesafede olmasından ve çizdiğim yeşil hatlara değmemesinden anlayabiliriz.

**Burada kutu modelini şema halinde göstereceğim:**



**Kutu modeli margin, padding, border ve içerikten oluşur.**

* **İçerik(content):** Elementin içinde olan resim, ses ya da yazıdır.
* **Padding:** İçerik ile border arasında olan boşluk.
* **Border:** Padding ile margin'i ayıran sınırdır.
* **Margin:** Kutunun diğer kutularla arasındaki mesafeyi ayarlayan boşluk.

Yukarıda saydığım kavramların dışında kutunun boy ve uzunluğunu belirleyen width ve height kavramları vardır. Bunları da alıştırmalarımızda göstermeye çalışacağım.

# Width ve Height

width ve height özellikleri ile elementin, yani lego parçamızın boy ve genişliğini ayarlayabiliriz. Bu özellikleri biz yazmadığımız takdirde tarayıcı kendisi ayarlar ve yapacağımız web uygulamasında istenmeyen sonuçlara neden olur :)Bu özellikleri px, em veya % ile kullanabiliriz. % ile kullandığımız zaman elementimiz hangi elementin içindeyse ona göre oranlanır. Yani 800px boyunda bir <div> tag'inin içinde bulunan bir elementin boyunu 50% olarak ayarlarsanız elementin boyu 400px olur.

Max-width özelliği ise bir elementin genişliğini sınırlamak istediğimizde kullanırız. Diyelim ki bir element tarayıcı ekranından fazla geniş veya uzun. **Bu durumda scrollbar(kaydırma çubuğu) çıkar.**

### **Alıştırma: Width ve Height Kullanımı**

<style>

p{

width: 0;

height: 0;

}

<style>

<p>

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

</p>

Bu kodda width ve height'i 0 yaptım. Paragraf tuhaf bir şekilde görünecektir. Bu alıştırmada yapmanız gereken şey paragrafın düzgün görünmesi açısından width ve height'i istediğiniz şekilde kullanın. Burada width ve heigth'i görmeniz için % ve px in ikisini de kullanmanızı tavsiye ediyoruz.

## Margin ve Padding

Margin

Margin özelliği ile elementin dışında boşluk oluşturmada kullanıldığını öğrenmiştik. Şimdi nasıl kullanıldığını açıklamaya çalışacağız.

margin özelliğini 50px yaptığınızda elementin dört yanında 50px boşluk oluşturulur. Element sağa doğru kayar. Negatif değerler de kullanılabilir. margin değerine -50px yazdığınız takdirde element sola doğru kayacaktır. Dört yanında değil de sadece bir yönde boşluk oluşturmak isteyebilirsiniz. Bunun için de özellikler vardır.

* margin-left: Soldan boşluk bırakır.
* margin-top: Yukarıdan boşluk bırakır.
* margin-bottom: Aşağıdan boşluk bırakır.
* margin-right: Sağdan boşluk bırakır.

Bu değerleri tek bir satırda ayarlamanız mümkündür.

margin: 100px 150px 60px 50px;

Bir elementi ortalamak için margin:auto değeri vermelisiniz. Bunun düzgün çalışması için de width özelliğinin de kullanılmasını öneririz. Böylelikle sağdan ve soldan eşit olarak ortalanmış olur. Aksi halde boyutu belli olmayan bir kutuyu ortalamayabilir.

### **Padding**

Şimdi gelelim padding özelliğine. Elementin içindeki yazıların **çerçeve**(border) ile mesafesini padding ile ayarlarız. margin özelliğinde olduğu gibi padding:50px yaptığımızda içeriğin her tarafı için 50px boşluk bırakıyoruz. Belirli bir tarafta padding bırakmak için;

* padding-top: içeriğin yukarısında boşluk bırakır.
* padding-bottom: içeriğin aşağısında boşluk bırakır.
* padding-left: içeriğin solunda boşluk bırakır.
* padding-right: içeriğin sağında boşluk bırakır.

Bunları tek tek tanımlamak yerine tek bir satırda belirlemek mümkündür.

padding: 10px 20px 30px 40px;

/\* Burada üst boşluk 10px,sağ boşluk 20px,alt boşluk 30px ve sol boşluk 40px dir.\*/

padding: 10px 20px 30px;

/\* Üst boşluk 10px,sağ ve sol boşluk 20px,alt boşluk 30px dir. \*/

padding: 10px 20px;

/\* üst ve alt boşluk 10px,sağ ve sol boşluk 20px \*/