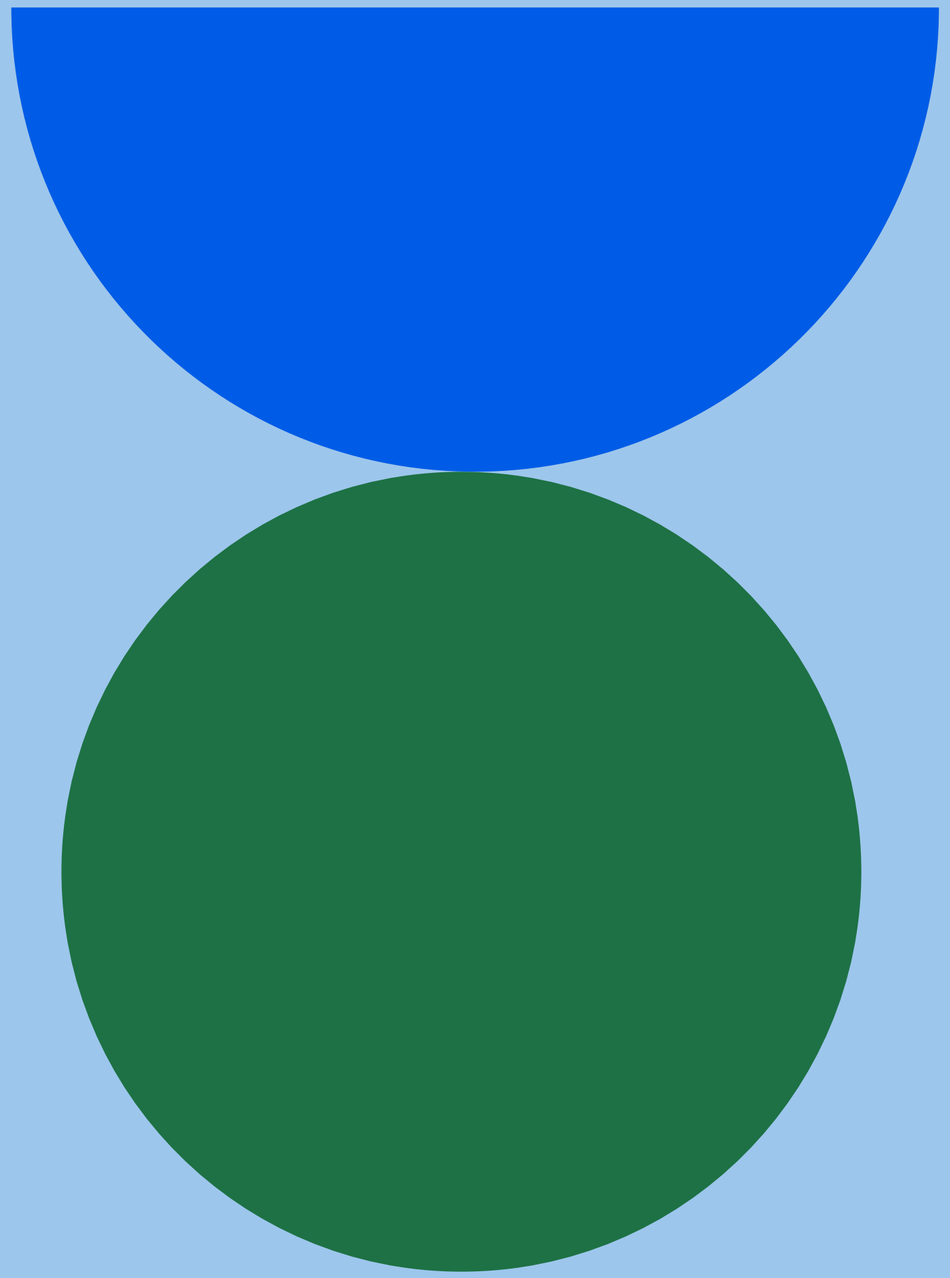
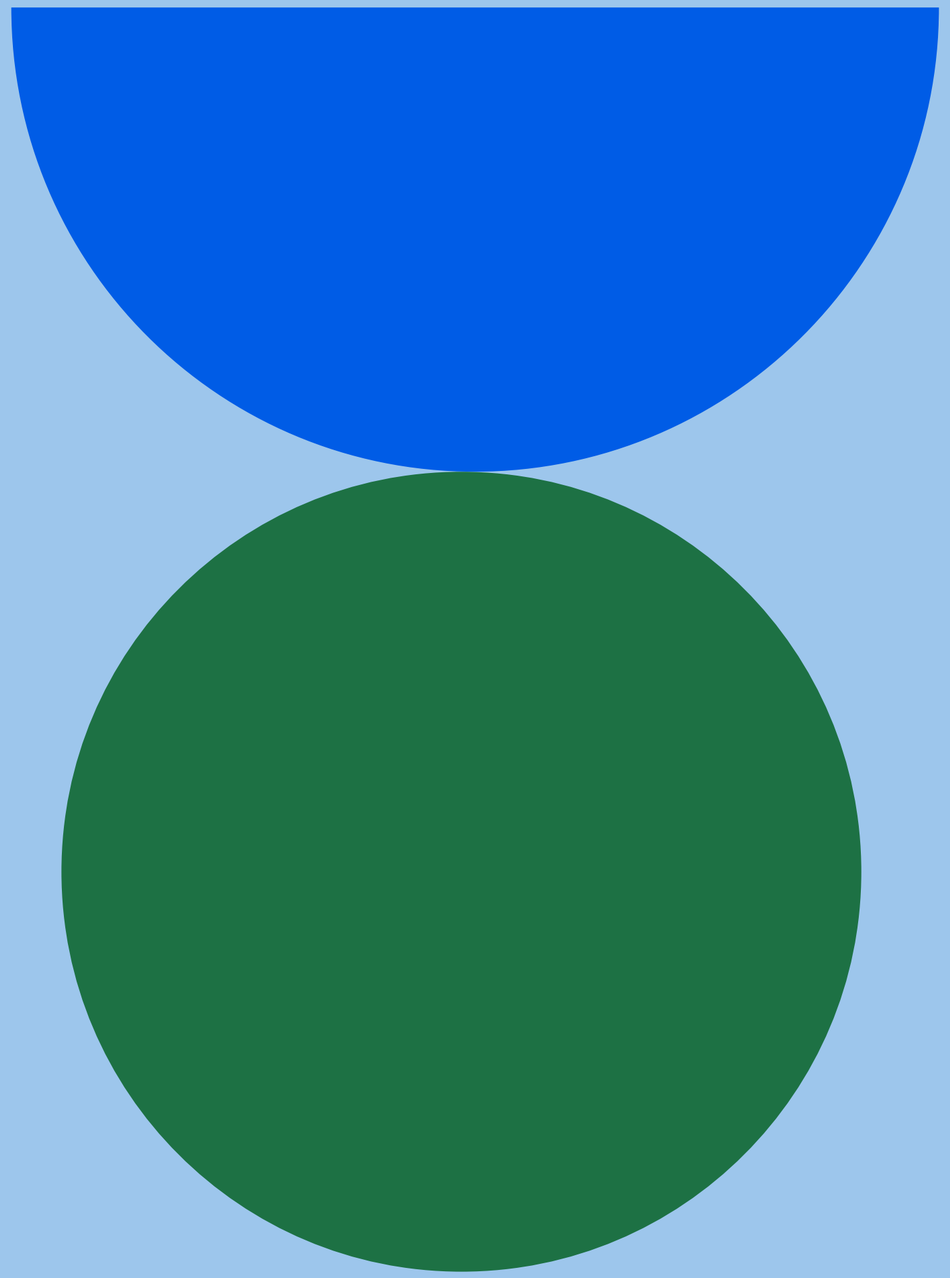


JavaScript'te closure nedir ve nasıl/neden kullanılır?

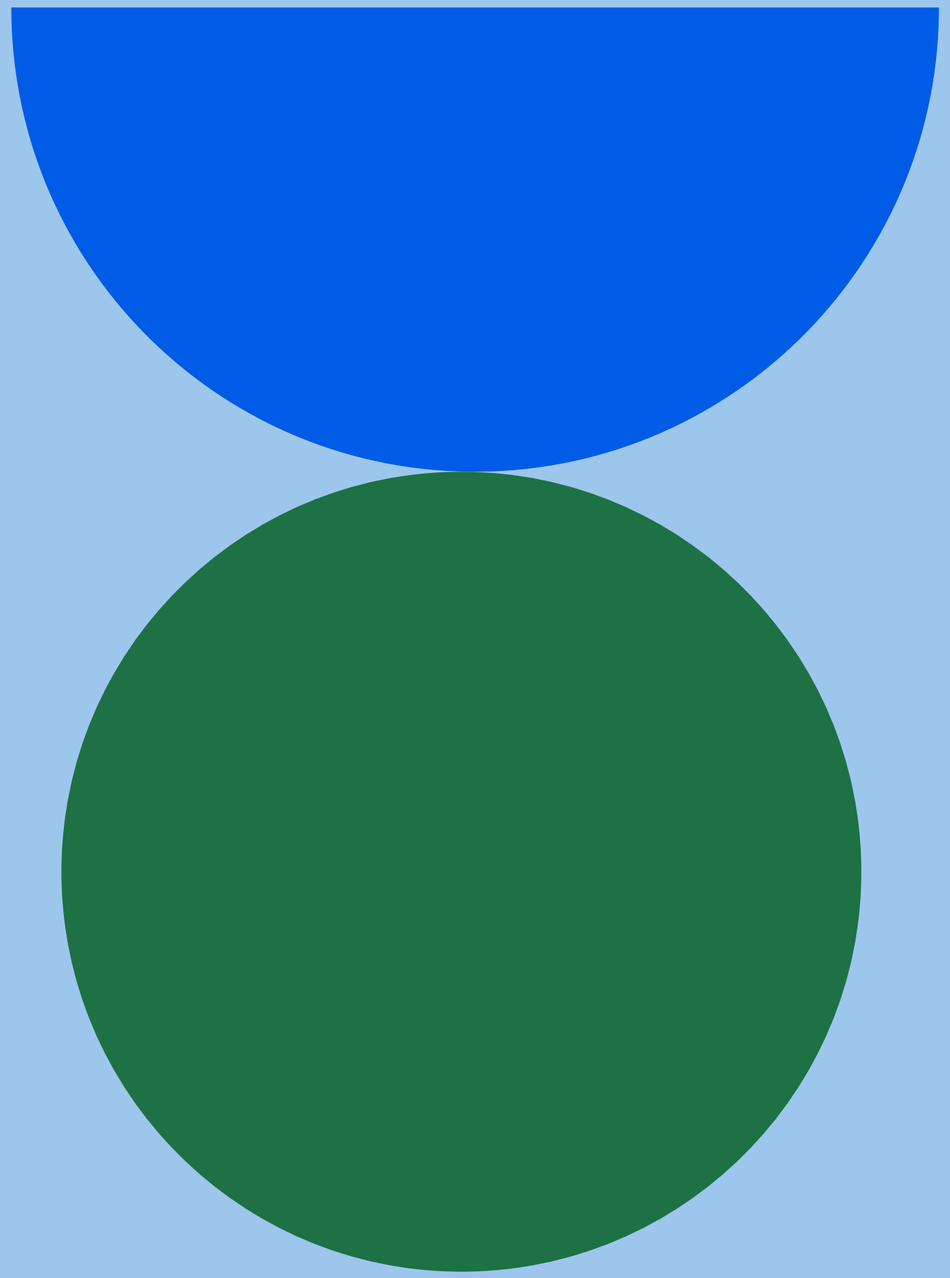
JavaScript'te closure (kapanış), bir iç fonksiyonun, dış fonksiyonun kapsamındaki değişkenleri ve parametreleri yakalayabilmesi anlamına gelir. Bu, iç fonksiyonun kapsamı dış fonksiyonun kapsamına dahil edildiği için mümkündür.



JavaScript'te closure nedir? (Lexical Environment)

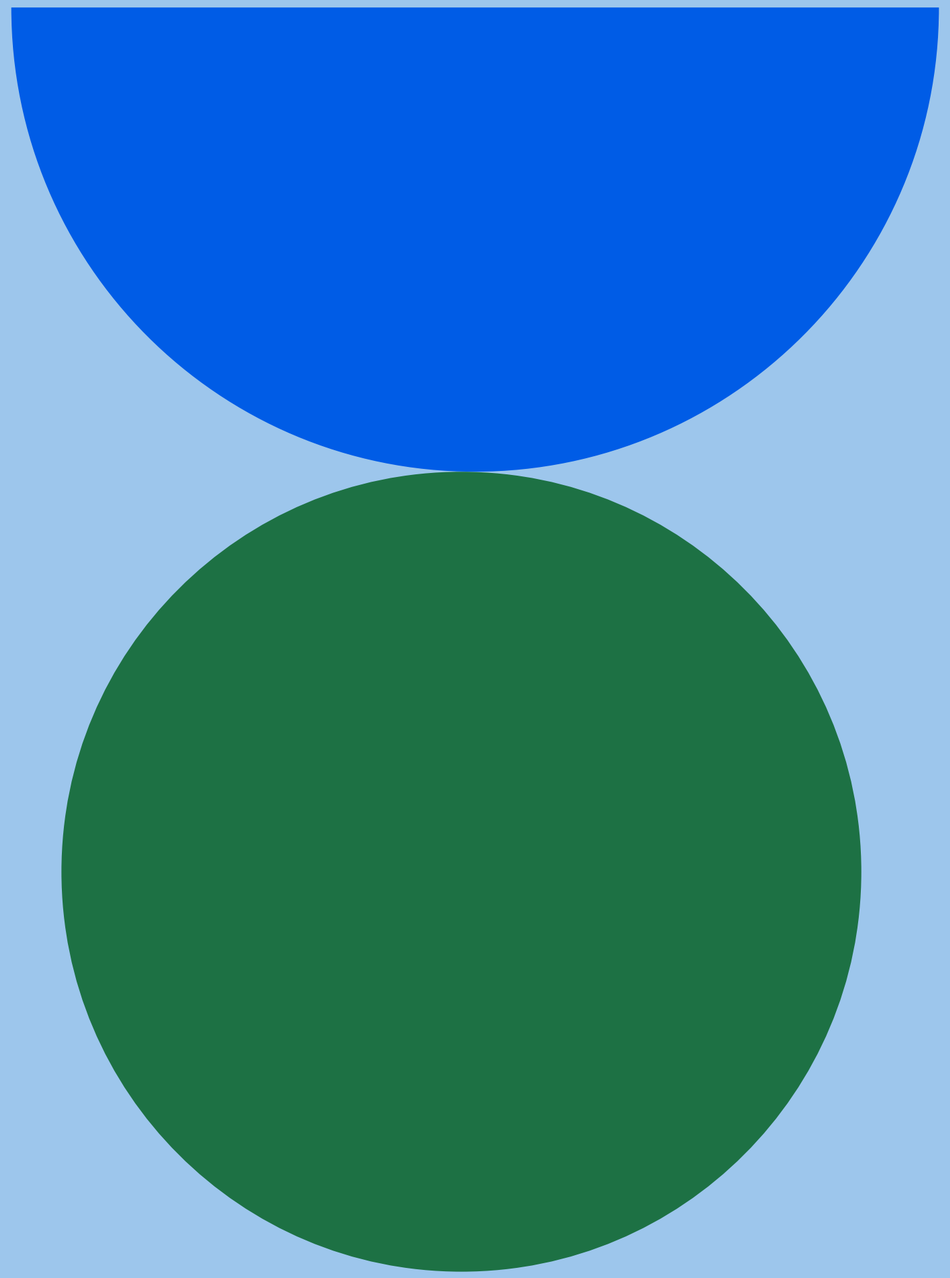
- JavaScript'te bir fonksiyonun bir iç fonksiyonu dışında bir kapsama (scope) ile birlikte kullanılmasıdır.

- Bu, bir fonksiyonun, iç fonksiyonun kapsamındaki değişkenlere erişebileceği anlamına gelir, ancak bu değişkenlerin dışarıdan erişilebilir olmaması anlamına gelir.



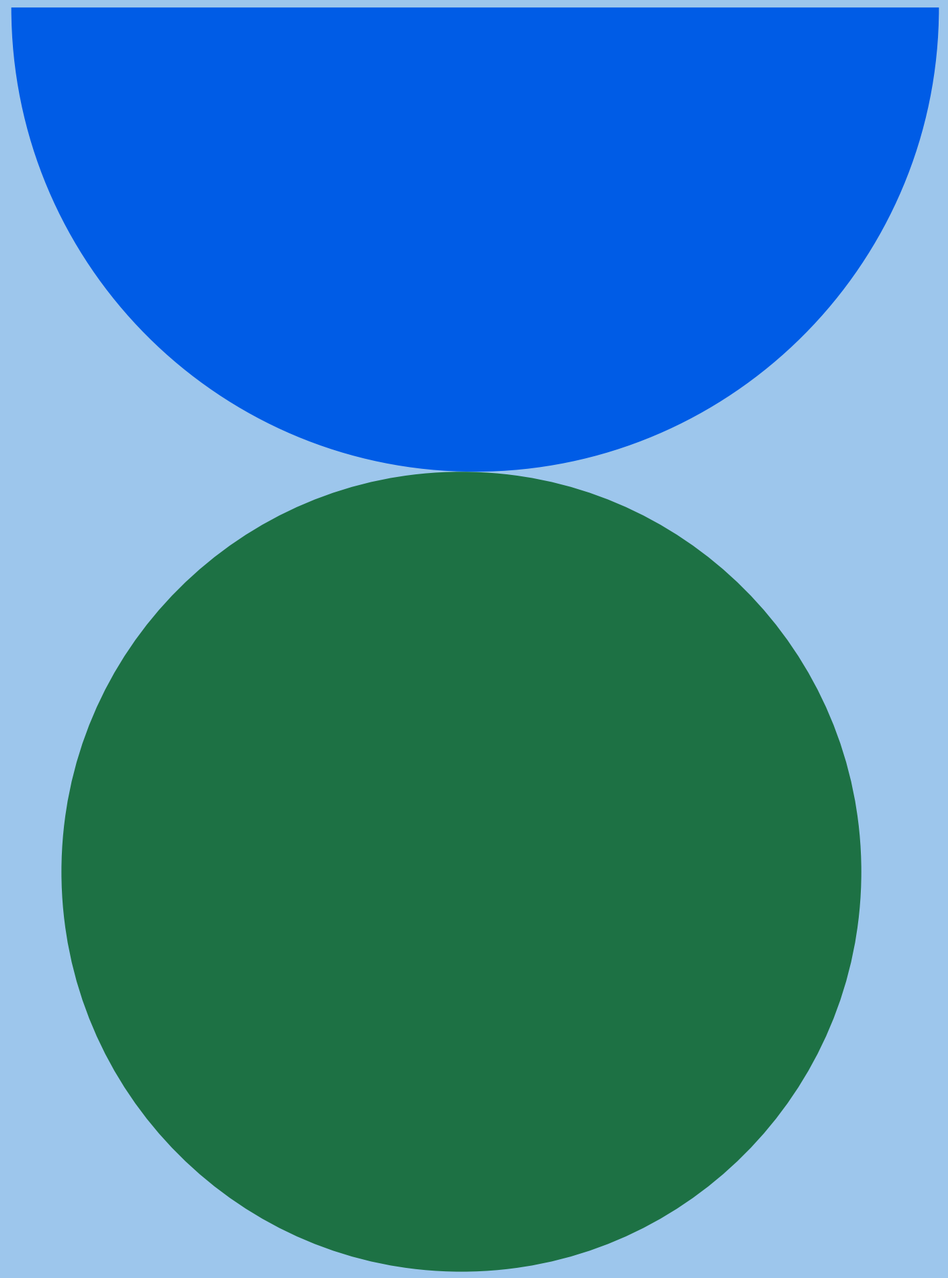
JavaScript'te closure nedir? (Lexical Environment)

- Closure, fonksiyonlara özgü bir özelliktir ve fonksiyonları daha esnek ve güçlü hale getirir.
- Bir fonksiyon içinde tanımlanan bir değişken, fonksiyonun çalışması bittikten sonra da hafızada tutulur.
- Dışarıdan erişilemez, ancak iç fonksiyonlar bu değişkenlere erişebilir.



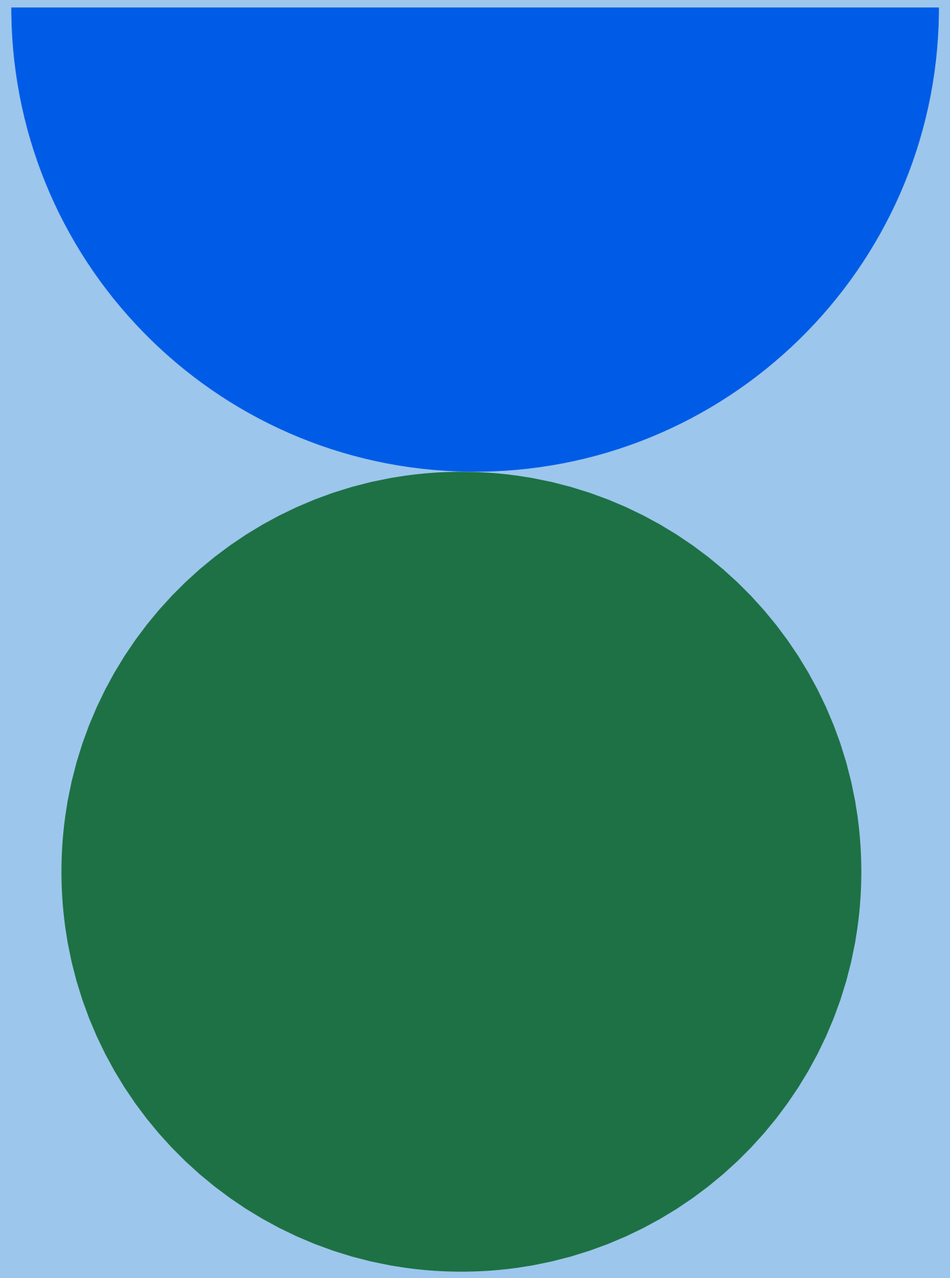
JavaScript'te closure nedir?(Lexical Environment)

- Bunun temelinde, JavaScript'in Lexical Environment (Sözdizimi Çevresi) yapısı yatmaktadır.
- Lexical Environment, bir fonksiyonun çalıştığı ortamı temsil eder ve bu ortamda tanımlanan değişkenler ve fonksiyonlar dahil olmak üzere tüm yerel değişkenleri içerir.
- Bir iç fonksiyonun kapsamı, Lexical Environment yapısı nedeniyle, ana fonksiyonun kapsamına erişebilir.



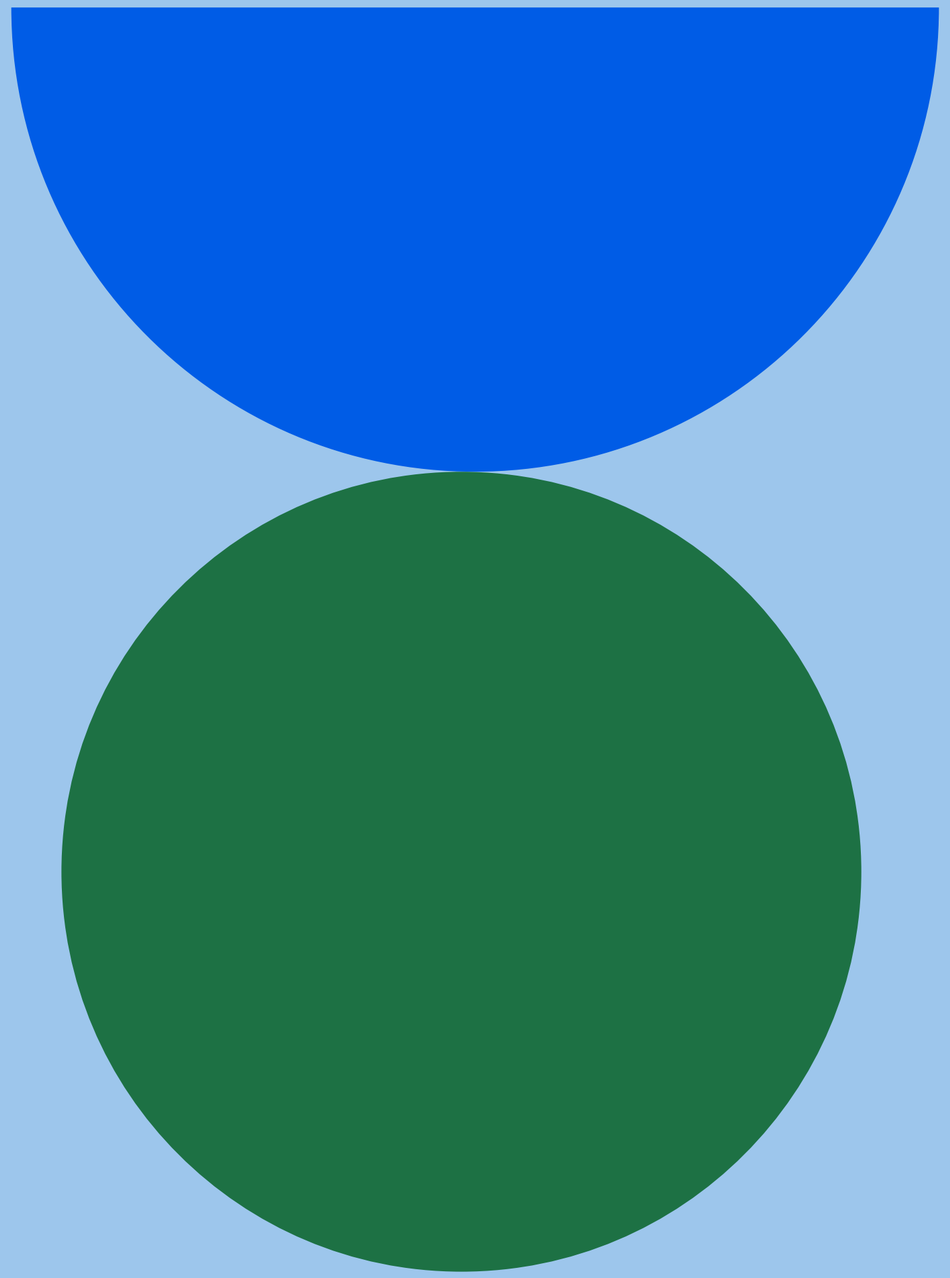
JavaScript'te closure nedir? (Lexical Environment)

- Closure, fonksiyonların bir özellik olarak kullanılabildiği birçok farklı senaryoda kullanılabilir.
- Örneğin, bir fonksiyonun bir iç fonksiyonu aracılığıyla bir değişkeni güncellemesi veya bir olay dinleyicisi olarak kullanılması gibi durumlarda kullanılabilir



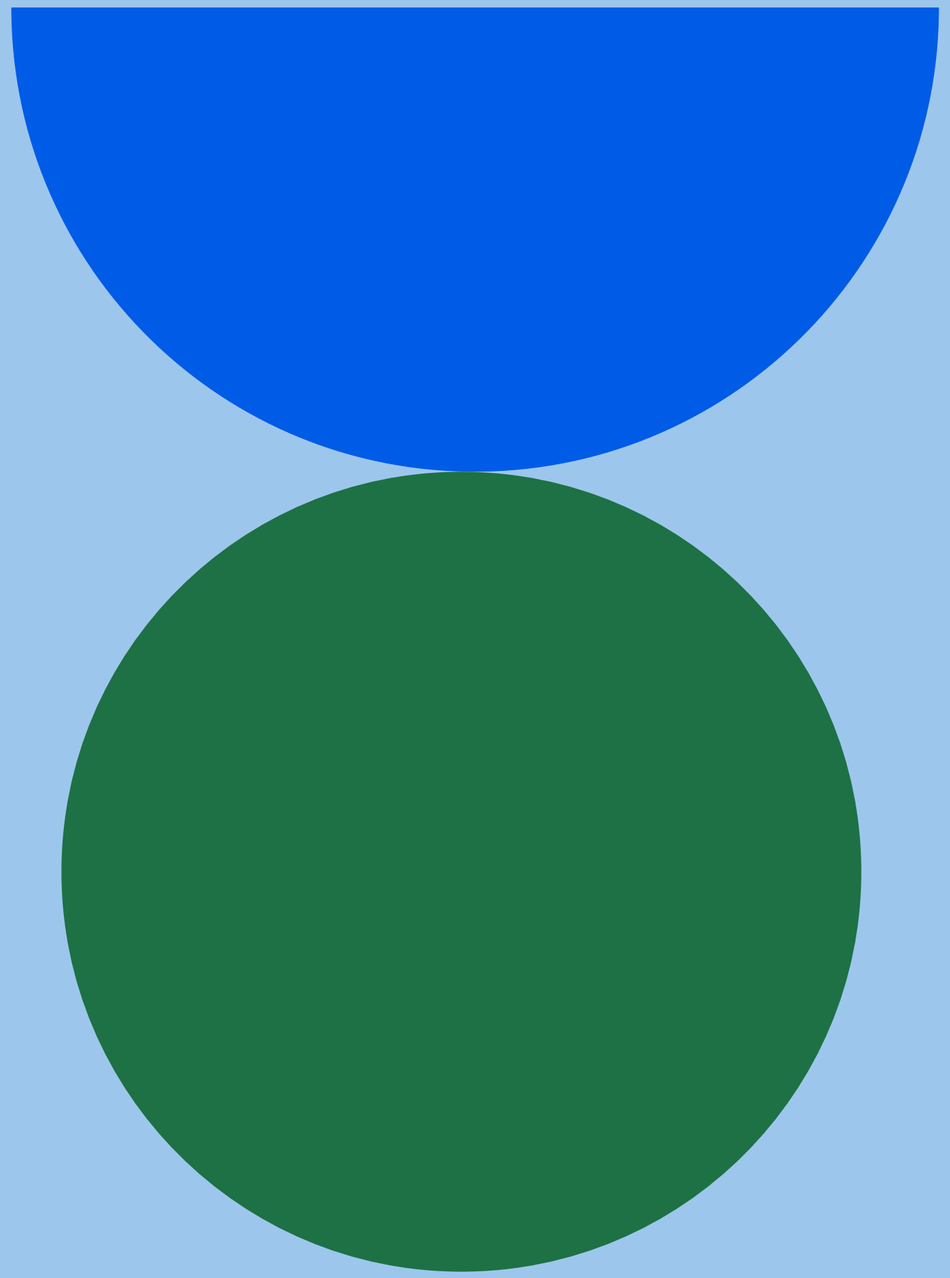
JavaScript'te closure nedir?(Lexical Environment)

- İç fonksiyonun local scope'u ile birlikte global scope da JavaScript'te kullanılan kapsamlardan biridir.
- Global scope, herhangi bir fonksiyonun içinde veya herhangi bir blokun içinde değil, en dışarıdaki kapsamı ifade eder. Global scope'ta tanımlanan değişkenlere, tüm fonksiyonlar ve bloklar tarafından erişilebilir.



JavaScript'te closure nedir?(Lexical Environment)

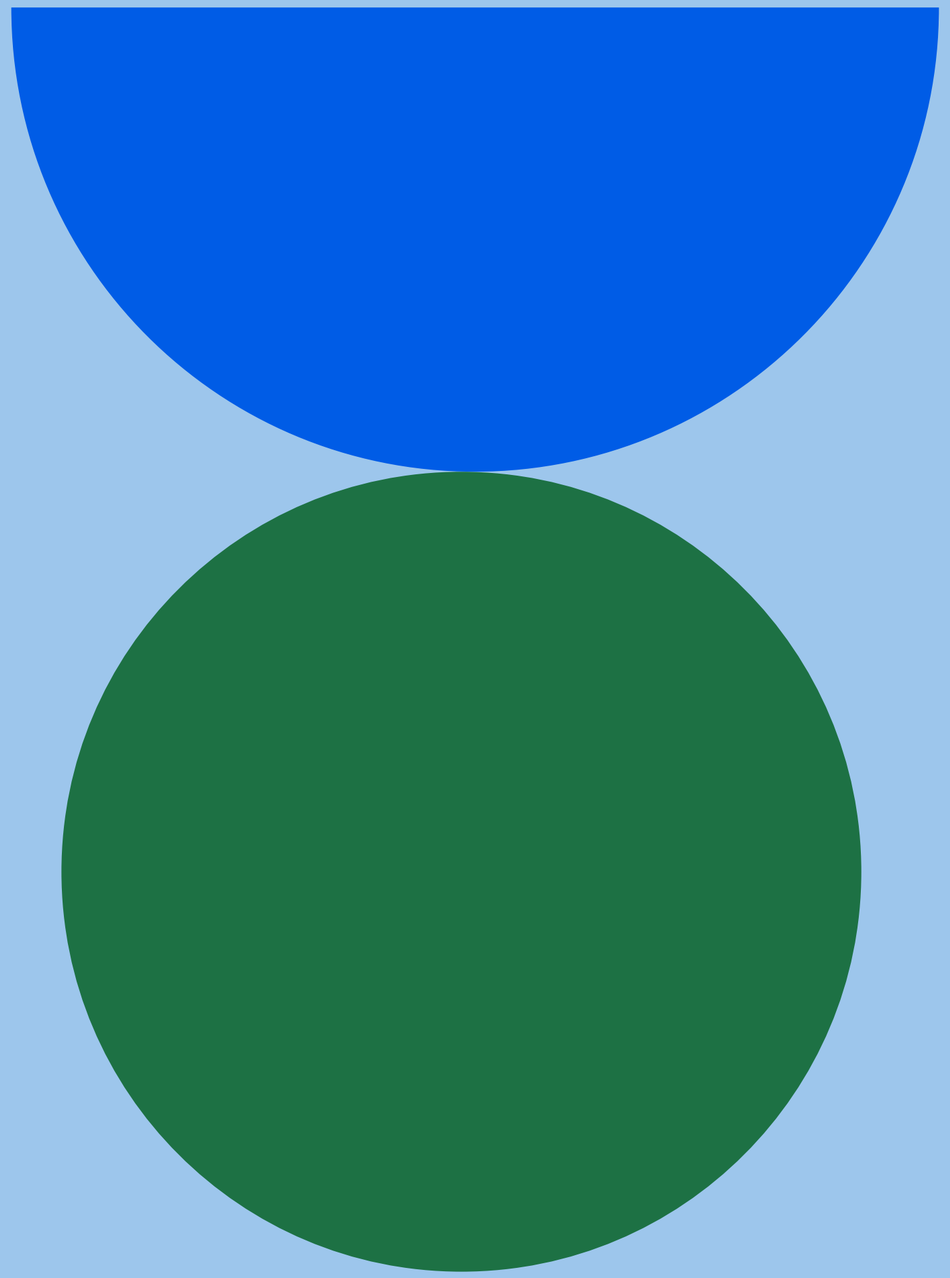
- Bir fonksiyon içinde tanımlanan değişkenler local scope'ta, yani fonksiyonun kapsamında kalırken, global scope'ta tanımlanan değişkenler tüm kod bloklarında ve fonksiyonlarda erişilebilir. Ancak, global değişkenlerin kullanımı, kodun anlaşılabilirliğini ve bakımını zorlaştırabilir ve global değişkenlerin yanlışlıkla başka yerlerde değiştirilmesine neden olabilir. Bu nedenle, global değişken kullanımı, mümkün olduğunca sınırlanmalı ve yerel değişkenler tercih edilmelidir.



Closure kullanmanın birkaç nedeni şunlar olabilir:

- Gizlilik: Closure, bir fonksiyonun kapsamındaki değişkenleri ve fonksiyonları dışarıdan erişilemez hale getirir. Bu nedenle, closure, özel bir bilgi veya işlevselliği saklamak için kullanılabilir. Bu, bir değişkenin veya fonksiyonun belirli bir kapsamda saklanması ve yalnızca ihtiyaç duyulduğunda kullanılabilmesi anlamına gelir.

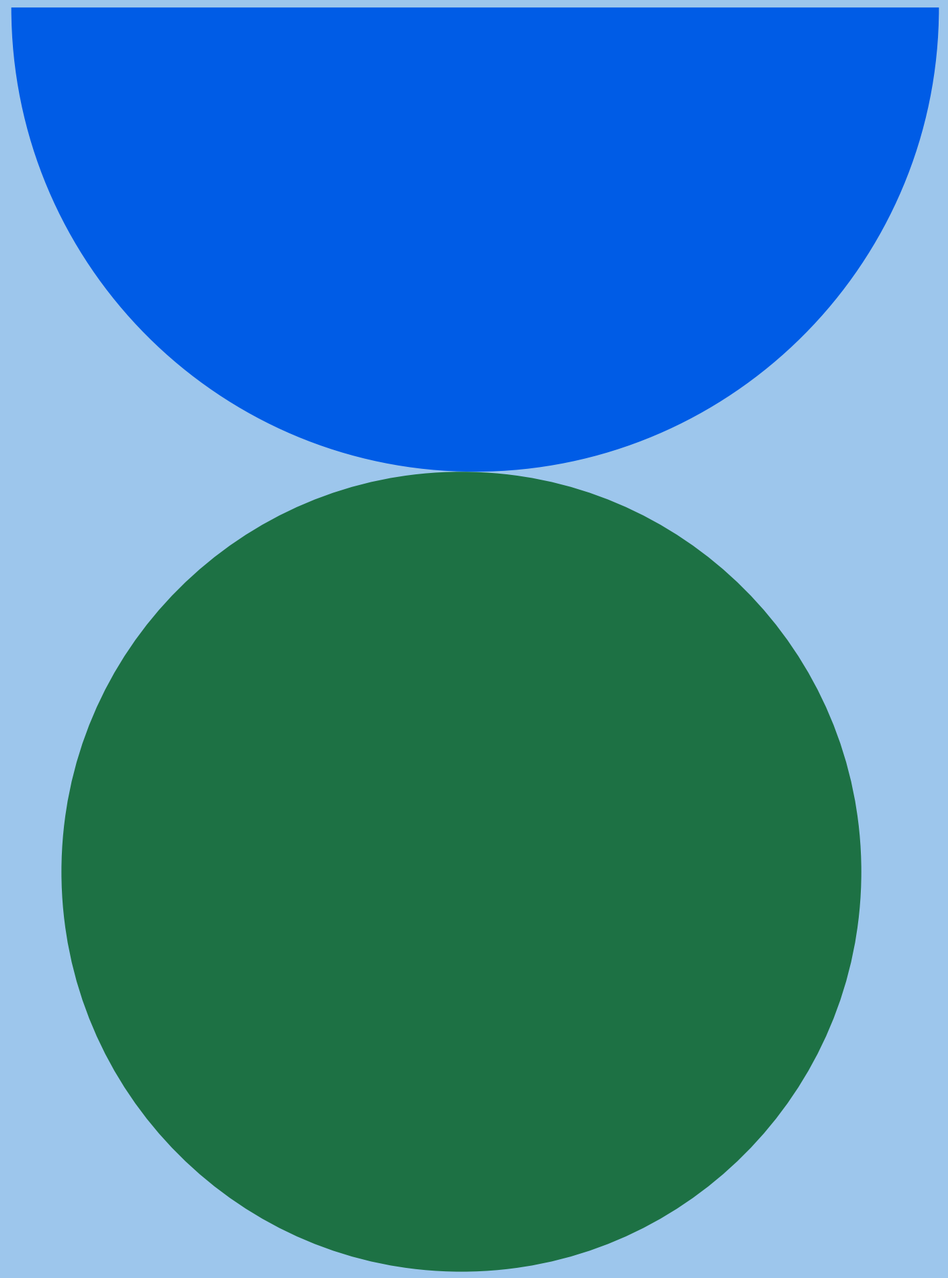
- Kapsam yönetimi: Closure, değişkenlerin ömrünü yönetmek için kullanılabilir. Örneğin, bir değişkenin ömrünü yalnızca bir fonksiyon çalıştığı sürece sınırlamak mümkündür.



Closure kullanmanın birkaç nedeni şunlar olabilir:

- Fonksiyonlar arası veri paylaşımı: Closure, bir fonksiyondan diğerine veri aktarmak için kullanılabilir. Bir fonksiyondan diğerine veri aktarmanın başka yolları da vardır, ancak closure, bu işlemi güvenli ve ölçeklenebilir bir şekilde yapmak için kullanılabilir.

- Callback fonksiyonları: Callback fonksiyonları, bir fonksiyonun başka bir fonksiyona parametre olarak geçirilmesi ve çağrılmasıdır. Callback fonksiyonlarının kullanımı, closure'ların kullanımıyla birleştirildiğinde, asenkron programlama için çok güçlü bir araç haline gelebilir.



Closure kullanmanın birkaç nedeni şunlar olabilir:

- Bellek yönetimi: Closure, gereksiz bellek tüketimini önlemek için kullanılabilir. Örneğin, bir değişkenin kullanımının tamamlanması durumunda bellekteki yerini temizlemek için closure kullanılabilir.

- Closure'lar, JavaScript'in daha gelişmiş ve güçlü programlama özelliklerini kullanarak daha ölçeklenebilir ve yönetilebilir kodlar oluşturmanıza olanak tanır.

Bu örnekte, greeting fonksiyonu, name adlı bir parametre alır ve message adlı bir değişken tanımlar. Fonksiyon, içindeki displayName fonksiyonunu döndürür. displayName fonksiyonu, dış fonksiyonun kapsamındaki name ve message değişkenlerine erişebilir ve bu değişkenleri kullanarak bir mesaj yazdırır.
Bu örnekte closure kullanarak, greeting fonksiyonu her çağrıldığında farklı bir name parametresi alabilir ve her seferinde farklı bir sayHello fonksiyonu döndürebilir. Bu, fonksiyonların özelleştirilmesine ve tekrar kullanılmasına olanak tanır.

```
function greeting(name) {
  var message = "Hello, ";

  function displayName() {
    console.log(message + name);
  }

  return displayName;
}

var sayHello = greeting("John");
sayHello(); // Hello, John
```