Cevaplar

**1.** Callback cehennemi iç içe callback fonksiyonlarının (asenkron) fonksiyonların çağırılmasıdır. Örneğin, içiçe http request ya da bir dosya okuma olduktan sonra http request kullanılması. Bundan kurtulmak için async/await fonksiyonu kullanırdım. Şöyle ki,

Bir dosyayı okuyup içindeki verileri almak için bir değişkene “await fs. read(dosya ismi)”. Dosya okumanın ardından bir http requestin atılması gerektiğinde başka bir değişken içine “await http.get(url)” şeklinde bir çözüm yapılabilir.

Örneğin,

Dosyanın içinde bir url var. O url’yi okuduktan sonra http request atmam gerektiğinde ilk önce “await fs. read(dosya ismi)” sonra içindeki veriyi okuduktan sonra o veri üzerinden başka bir request işlemi yapacaksam “await http.get(url)” kullanırım.

**2.** Node js de Event loop denilen bir kavram var. Bu kavramda, her gelen istek asekron olarak gelir. Bu asekron istekler herhangi bir başka işlemi beklemeden gelen isteğe göre kullanıcının isteğini yerine getirir.

**3.** Global değişken kullanmamız gerektiğinde mutlaka user’a özel bilgilerin tutulmaması gerekir. Bunun için session kullanılmalıdır. Ayrıca, açık yaratılmayacak şekilde global verilerin kullanılması gerekir. Genellikle global verileri bir kütüphaneyi require etmek için kullanmalıyız.

**4.** let parent = {

func2: function() {

return new Promise(function(resolve, reject){

return resolve(1);

})

} ,

func1: function() {

**let vm = this;**

return new Promise(function(resolve, reject){

**vm**.func2().then((count) => {

console.log(count + 1)

})

})

}

}

parent.func1();

Açıklama: Yukardaki kodta yaptığım gibi Promise’in içinde object’in “this” ögesini görmediği için, Promise’e girmeden önce bir değişken içerisine object’in “this” ögesini atadım. Ve promise içinde dışarıda “vm” diye tanımladığım değişken içerisindeki “this” yapısına ulaşmış oldum.