Object ve Class kavramları

Derste class kavramına doğrudan kurucu function (constructor()) üzerinden giriş yapmak özellikle yeni başlayan arkadaşların kafasını oldukça karıştırdı diye düşünüyorum. İşte bu nedenle böyle bir yazı ile kavramları daha anlaşılır bir hale getirme ihtiyacı duydum.

Aslında class yapıları daha çok çoklu veriler, özellikle de veritabanı uygulamalarında kullanılıyor. Yani Back end yazılımlarda. Java script front end bir yazılım olduğundan olsa gerek şimdiye kadar pek ihtiyaç duyulmamış olsa gerek sonradan bir güncelleme ile geldi. Tabi bu javascriptin daha önce nesne yönelimli bir yazılım olmadığı anlamına gelmiyor. Java script hep nesne yönelimli bir yazılım diliydi. Ancak yapısında var olan bu özelliği kullanıcılara açmamıştı belki. Ya da ihtiyaç duymamıştı.

Bilindiği gibi, içerisinde farklı tip ve yapılarda değişkenler barındırabilen yapılara biz object diyoruz. Peki neler? Örneğin sayı, metin, array, function, hatta başka bir object.

```
Let obj = {
      name: "Erdoğan",
      surname: "Bülbül",
      yas: 53,
      isimYaz(){ … },
}
```

Object içerisinde tanımladığımız değişkenlere property, function() lara ise metod adı veriyoruz.

Peki class ne oluyor?

Diyelim ki alçıdan heykel yapan bir atölyeniz var. Heykel bizim nesnemiz, alçı ise hammaddemiz. Yani alçı veriyi, heykel bilgiyi temsil ediyor. Veri ve bilgi kavramlarını daha iyi anlamak için: https://erbull.github.io/programming/datastructure.html.

Eğer tek bir heykel sipariş almışsanız kalıpla uğraşmanıza gerek yok. Bunun için harcayacağınız enerjiyi doğrudan heykeli yapmaya harcarsınız. Yani object bizim işimizi görür. Peki aynı heykelden birlerce yapmanız gerekseydi?

Bu durumda tabiki mantıklı olan alçıyı, yani veriyi içerisine döküp heykel şekline dönüştürebileceğimiz bir kalıp yapmak olurdu. Yani class. Bu sayede istediğimiz kadar heykeli zahmetsizce üretebiliriz.

Yani bir veriyi, daha doğrusu bilgiyi sadece bir defa kullanacaksanız, ve daha sonra o bilginin hafızada kalmasının bir gereği yoksa bize object yeterli. Hatta o object i içerisindeki verileri değiştirip başka yerde kullanmamız da mümkün. Ama bu durumda tabi ki obje içerisindeki eski verilere veda etmemiz gerekecek. Çünkü artık onların yerini yeni değerler aldı.

Ama siz istiyorsunuz ki, bu object ye benim başka yerlerde de ihtiyacım var ama, içerisinde bulunan verilerle henüz işim bitmedi. Hala onlara ihtiyacım var ve ileride de olacak. Yani bu object nimn aynısından lazım ama eskisi de kaybolmasın. İşte burada class devreye giriyor. Yani istediğim sayıda object üretip kullanabileceğim bir veri kalıbı. Yapı hemen hemen object ile aynı. Yani hemen hemen. Birkaç küçük fark var tabi m onlara sonra geleceğiz.

```
class person {
      name;
      surname;
      age;
      isimYaz() { … };
}
```

Class ımız hazır. Artık bundan istediğimiz kadar object üretip içini doldurabiliriz. Ve hiçbir object'nin içeriğinin diğerine etkisi olmaz. Yani hepsi ayrı birer heykel. Birini kırmızıya, birini maviye, bir diğerini pembeye boyayabiliriz. Yapı olarak aynılar, yani aynı özelliklere sahipler ama, özellikleri birbirinden farklı.

```
Let obj1 = new person();
obj1.name = "Ali";
obj1.surname = "Duymaz";
obj1.age = 28;

let obj2 = new person();
obj1.name = "Veli";
obj1.surname = "Görmez";
obj1.age = 35;
```

Ama şu an class ımızda bir sorun var. Daha doğrusu bir tür eksik. C# dilinde class tan bir object üretirken hemen devamında süslü parantez içerisinde propertylerin değerlerini atayabiliyoruz. Gördüğüm kadarıyla JavaScript buna izin vermiyor. Yani object oluşturduktan sonra içerisindeki özelliklere, yani değişkenlere tek tek atama yapmak zorundayız. Peki bu sorunu nasıl çözebiliriz?

Amacım elbette buradan yavaş yavaş condtructor() (yapıcı) function olayına gelmek. Ama önce bahsettiğimiz sorunu çözmenin bir yolunu bulalım. Örneğin bu sorunu, class içerisine bir function yazarak halletmemiz mümkün mü? Mesela şöyle bir function:

```
class person {
      name;
      surname;
            age;
            degerata(ad, soyad, yas){
            this.name = ad;
            this.surname= soyad;
            this.age = yas
      }
}
```

Gördüğümüz gibi, dışarıdan ad, soyad, yas isminde üç değer alan bir function yazdık ve aldığımız bu değerleri class ımızın name, surname ve yas property lerine atadık. Artık şöyle yapabiliriz.

let obj = new person();

obj.degerata("Erdoğan","Bülbül",53);

Hatta değişkenleri önceden tanımlamak yerine bu functionda da tanımlayabiliriz. JavaScript bu yönden esnek bir dil. Eğer var olmayan bir değişkene, özelliğe atıf yaparsanız kendisi oluşturuyor.

```
class person {
            degerata(ad, soyad, yas){
            this.name = ad;
            this.surname= soyad;
            this.age = yas
      }
}
```

Gördüğünüz gibi, functionumuz aldığı değerleri atamak için this nesnesi, yanı class ımızın ana gövdesi üzerinde name isimli bir değişken arayacak, bulamazsa oluşturacak ve ad parametresi ile aldığı değeri bu değişkene atayacak. (Değişkenden kastımızın property (özellik) olduğunu artık biliyoruz. Class ana gövdesi içerisinde tanımlanan değişkenlere bir property diyoruz.)

Ama yine de bu işlem çok uzun oldu, yok mu daha kolayı diyecek kadar tembelseniz, bu işlemi iki adımda (obje oluştur, değerleri ata) değil de tek adımda yapmanın bir çaresini de bulabiliriz. Yani bu function ikinci bir komut yazmama gerek kalmadan –leb demeden leblebiyi anlasa- da beni yormadan kendisi çalışsa güzel olmaz mıydı?

Bunun için, class nesnesinin altyapısında ön tanımlı bir function olan constructor() den faydalanacağız. Bu function, diğer functionlardan biraz farklı. Yapı olarak aynı, bildiğimiz sıradan bir functiondan öte bir şey değil, kafanız karışmasın. Tek farkı, object oluşturulurken bir defaya mahsus kendiliğinden çalışması. Bu nedenle kurucu (yapıcı) function olarak ta bilinir. Yapmamız gereken tek şey, degerata() functionunun adını constructor() olarak değiştirmek. İşte oldu bitti.

```
class person {
      constructor(ad, soyad, yas){
            this.name = ad;
            this.surname= soyad;
            this.age = yas
      }
}
```

Artık bir object oluştururken içerisindeki değişkenlerin değerlerini de atayabiliriz. (Tebbel işi)

```
let obj = new person("Erdoğan", "Bülbül", 53)
```