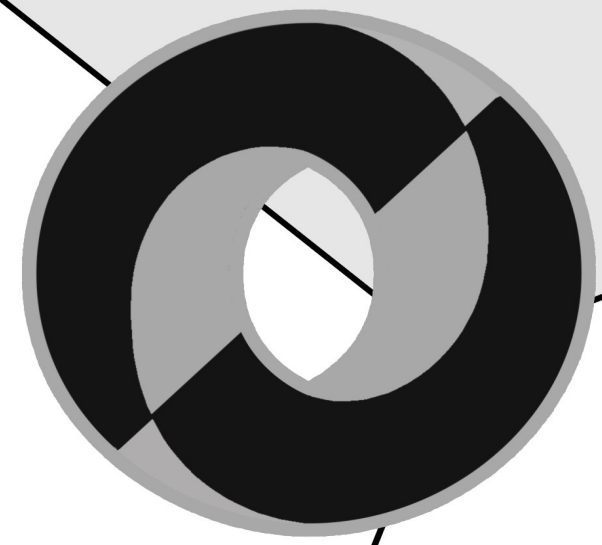


JSON

JavaScript Object Notation

```
{
"book":{
"name" : "JSON",
"edition": 1.0,
"author": "Emre Can ÖZTAŞ"
        }
}
```

Kapak Tasarımı: Samet KÖROĞLU

JSON

Book Version: 1.0

Yazan: Emre Can ÖZTAŞ

Önsöz

Merhaba, ben Emre Can ÖZTAŞ. Bu ve bundan önceki; Simple JavaScript ve Simple HTML5 kitaplarının yazarıyım. Yani bu benim üçüncü kitabım. Aslında kitap demek ne kadar doğru bilmiyorum ama e-book desek daha doğru olur, herhalde. Çünkü bu kitapların hiç birinin basımının yapılması veya herhangi bir ticari kaygı gütmedim. Belki birilerine bir faydam dokunur diye yazıp e-book haline getiriyorum. Sonrada bu kitapları paylaşıma açıyorum.

Daha önceki kitaplarımda da belirtmiştim. Benim bu işten zerre kadar kazancım yok. Benim tek amacım insanlara yardımcı olabilmek. En büyük umudum ve mutluluğum; belki birileri yazdıklarımdan birşeyler öğrenecek ve bu öğrendikleriyle parasını kazanıp evine ekmek götürecek. Yani kazandığı parada ya da ailesinin geçimini temin etmede belki birilerine yardımcı olacağım. Öyleyse ne mutlu bana.

Kaç yaşına gelmiş, vatana ve millete zerre kadar faydası olmayan, dilleri zehirli ok ve bu dilleriyle hep birilerini kandırma, hep kendi çıkarlarını gütme telaşında olan, kendi çıkarları adına yapamayacakları şey olmayan insanlarla dolu etrafım. Ben böyle olmak istemediğim için yazıyorum bu kitapları belki de. Bilmiyorum ama iyi bir şey için yazdığım kesin.

Yazdıklarımı hep sorgulayan, her fırsatta “neden yazıyorsun?”, “şimdi sırada ne var?”, “Benim hayatımı da yazsana” diye kendince espriler yapmaya çalışan havalı, kendini bir şey sanan fakat aslında boş ya da programlama dünyasındaki “null” bir değerde olan insanlarla da karşılaşıyorum. Unutmayın! Insanlar kendi yapamadıkları şeyleri sizlerin de yapamamasını isterler ve engel olmaya çalışırlar. O yüzden boş insanları fazla takmamalısınız.

Allah nasip eder belki ilerleyen yaşlarımı görürüm belki de göremem. Hayat kısa. Zaman daralıyor. Ama Allah nasip eder de görürsem şayet, geriye dönüp baktığımda neler yaptığımı hatırlamak adına da bu kitaplar benim için önemli olacaktır.

Güzel ülkemizin en büyük sıkıntısı: Türkçe kaynak. Türkçe'de yeterli kaynak yok. Bu yazdıklarım Türkçe kaynağa da destek olacaktır. Bunun da bilincindeyim. Lakin kitaplarımda bazı kavramların özellikle İngilizcesini kullanmaya çalışıyorum. Çünkü Türkçe'de bu kavramları karşılayabilecek bir tabir olmayabiliyor bazen.

Teşekkürler

Bu kitabın yazımında bana olumlu / olumsuz destek olan çevremdeki herkese, özellikle bana her konuda destek olan aileme teşekkür ederim. Siz olmasaydınız; gelişimimi tamamlayamaz, bügünkü eriştiğim noktaya erişemezdim. Iyi ki varsınız. Umarım hep var olursunuz.

Son olarak bu ve önceki iki kitabımın da kapak tasarımını yapan Samet KÖROĞLU'na teşekkürü bir borç bilirim.

İçindekiler

BÖLÜM 1:
JSON

Bu bölümde JSON yani JavaScript Object Notation hakkında genel bilgiler vereceğiz. Yani bir anlamda giriş bölümü olacaktır. Bu bölümde, JSON hakkındaki genel soruların cevaplarını bulmaya çalışacağız. Bir anlamda soru – cevap şeklinde geçecek bir bölüme hazırlıklı olun, baştan söyleyeyim.

Bu, kitabın ilk bölümü o yüzden “Ya Bismillah” diyerek JSON'a giriş yapalım.

1.1 JSON Nedir?

Öncelikle JSON nedir bu sorunun cevabını arayalım. JSON (JavaScript Object Notation), bir veri değişim formatı'dır. Peki veri değişim formatı nedir? Basit bir örnekle başlayalım. Günlük hayatta verilerimizi tutmak için veri tabanlarını kullanırız. Örneğin bir dinamik web sitemiz var olarak kabul edersek; bilgilerimizi illa ki bir veri tabanında saklamak zorundayız. Ayrıca bu veri tabanın da güvenliğini sağlamalıyız lakin bu kitabımızın konusu değil.

Bazı durumlarda elimizde olan veya veri tabanında saklanan bir takım verilerin paylaşılması veya daha genel bir anlamda yazılan programların birbirleriyle konuşması gerekir. Programlar da programlama dilleriyle yazıldığı için programlama dillerinin birbirleriyle konuşması veya anlaşması gerekir diyebiliriz. Örneğin Python framework'ü olan Django ile yazılmış bir web platformunun yeri geldiği zaman PHP ile yazılmış olan bir platformla konuşması gereken durumlar oluşabilir. Bu sadece 2 dil için geçerli değildir. Bütün programlama dilleri için geçerlidir. Işte böyle durumlarda devreye ara elemanlar girer daha genel bir tanımla veri taşıma formatları girer. XML'de bu ara elemanlardan birisidir. XML'i örnek vermemin sebebi bu zamana kadar kullanıla gelmiş genel bir veri değişim formatı olmasıdır.

JSON için XML'in alternatifi diyebiliriz. Tabiki JSON'un XML'e karşı avantajları ve dezavantajları var. Kitap boyunca bunlardan bahsedeceğiz. Şimdi veri taşıma formatına geri dönecek olursak; elimizdeki bazı verilerin diğer platformlarda kullanılabilmesi, taşınabilmesi veya değiştirilebilmesi için bazı ara elemanlara ihtiyacımız olacaktır demiştik. JSON'da XML gibi bir veri taşıma formatıdır. JSON ile belirttiğimiz işlemlerimizi gerçekleştirebiliriz.

JSON, bir dil değildir. Buraya dikkat edelim ve tekrar edelim: “JSON, bir dil değildir”. JSON, adından da anlaşılacağı üzere; JavaScript'ten türetilmiş bir yapıdır. Kendine has bir syntax'a (sözdizimi) sahiptir. Daha önce yazmış olduğum Simple JavaScript kitabının bir bölümünde kısaca JSON'a değinmiştim. Isterseniz, inceleyebilirsiniz.

JSON'un popülerliği giderek artmaktadır. Bunun nedeni; XML'e karşı bariz üstünlükleri var ve kullanımı kolay bir yapı. Ilk sorumuza cevap bulduğumuza göre bir diğer sorumuza geçelim.

1.2. Neden JSON?

XML varken neden JSON diye bir soruyla karşılabilirsiniz. Burada JSON'un XML'e karşı bariz üstünlükleri

var. Şimdi bunlardan bahsetmeye çalışalım.

Öncelikle JSON'un syntax yapısı XML'e göre daha basit ve okunabilirlik oranı yüksektir. Herhangi bir programlama diliyle az çok uğraşmış birisi JSON syntax'ını gördüğü zaman kolayca okuyabilir ve anlayabilir. Yani XML gibi karmaşık bir yapısı bulunmamaktadır. Aslında XML'de karmaşık bir yapıya sahip değildir fakat teferruatı çok dememiz daha doğru olacaktır.

JSON'un tercih edilmesinin bir diğer nedeni de `tags (etiketler)` olmamasıdır. Etiketlerden kastım daha az kod yazılmasıdır. Örneğin XML'de veriler etiketler arasına yazılır ve etiketsiz veri olmaz lakin JSON'da böyle birşey söz konusu değildir. Her veri içinde etiket yazılması çok tekrar yapılmasına neden olmaktadır. Bu dediklerimizi çok basit bir örnek üzerinde göstermeye çalışalım. Aynı veriyi hem XML hemde JSON formatında yazalım.

`XML Format:`

```
<uyelerTablosu>
   <uyeBilgi>
      <uyeAd>Emre Can</uyeAd>
      <uyeSoyad>OZTAS</uyeSoyad>
   </uyeBilgi>
   <uyeBilgi>
      <uyeAd>Hazal</uyeAd>
      <uyeSoyad>ONEY</uyeSoyad>
   </uyeBilgi>
</uyelerTablosu>
```

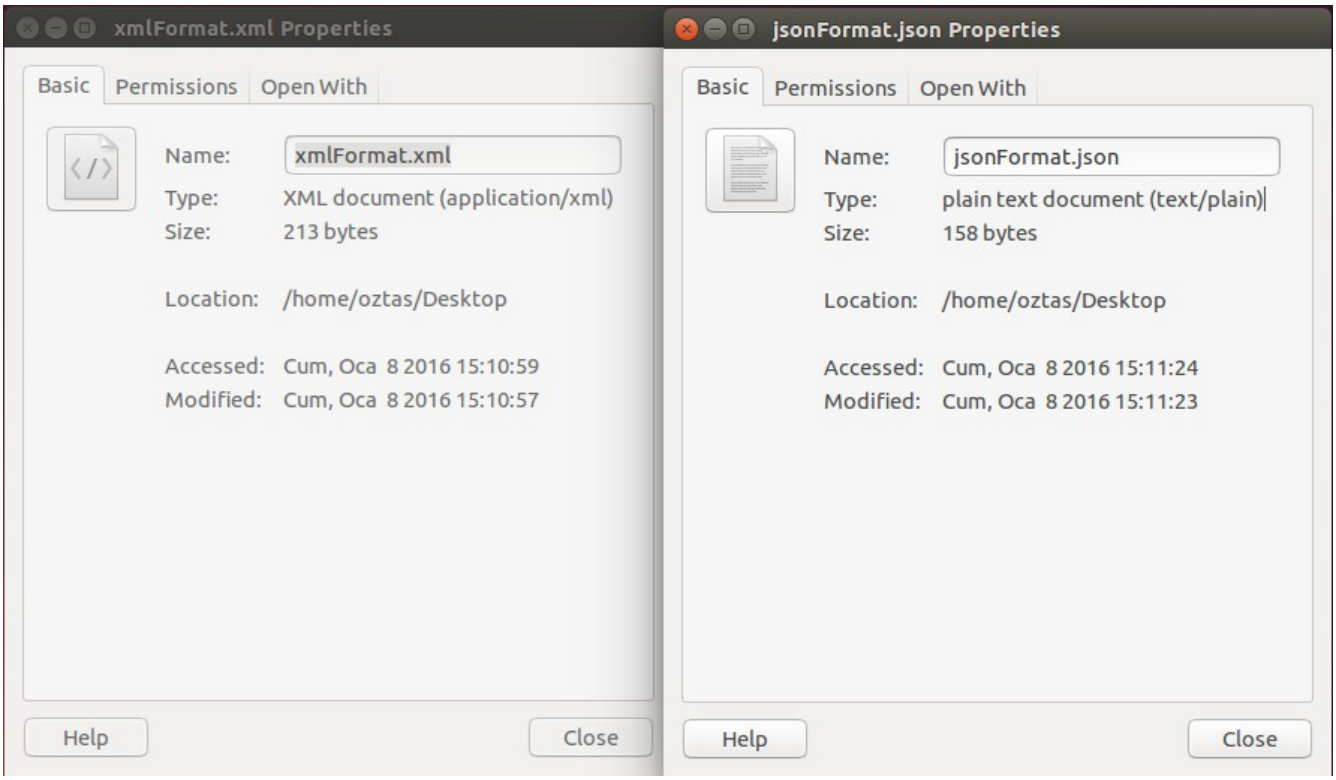
Yukarıdaki yazmış olduğumuz basit verimizi bir de JSON formatı ile yazalım.

`JSON Format:`

```
{
 "uyelerTablosu": [
    {
      "uyeAd" : "Emre Can",
      "uyeSoyad" : "OZTAS"
    },
    {
      "uyeAd" : "Hazal",
      "uyeSoyad" : "ONEY"
    }
 ]
}
```

Yukarıdaki her iki kullanımda da görüldüğü gibi JSON verileri açık ve temiz bir yapıya sahip. Şimdilik yukarıdaki gösterimleri bilmediğiniz için endişelenmeyin ilerleyen zamanlarda fazlasıyla detaya inecek ve JSON'u yakından tanımaya çalışacağız.

JSON syntax'ında etiketlerin bulunmayışı, JSON'ın dosya boyutunu oldukça azaltır. Yani XML'e göre daha az yer kaplar. Günümüzde verilerin taşınabilinmesi, güvenliği v.s önemli bir konu. Bir diğer önemli bir konuda bu veri dosyalarının boyutları. Şimdi basit bir test yapalım. Yukarıdaki yazmış olduğumuz; XML ve JSON yapıları ayrı ayrı olarak diske kaydedelim ve boyutlarına bakalım. XML formatındaki verimizi `.xml` uzantısıyla, JSON formatındaki verimizi `.json` uzantısıyla diske kaydedelim ve boyutlarına bakalım.



Yukarıdaki ekran alıntısında da görüldüğü gibi JSON veri formatı, XML veri formatına göre diskte daha az yer kaplamaktadır. Buradaki değerler şimdilik sizlere çok küçük gelebilir lakin bizim verdiğimiz sadece basit bir örnekti. Örneğin elinizde 1 milyon kayıt olduğunu ve bunların taşınması gerektiğini düşünürseniz o zaman buradaki boyutlar anlam kazanacaktır.

JSON programlama dillerinden bağımsız bir yapıdadır. Yani bir çok programlama dili ile kullanılabilir. Başta da belirttiğimiz gibi farklı programlama dillerinin birbirleriyle konuşması gerekir. Bu aradaki konuşmayı da bağımsız bir araçla yapmak gerekir. JSON da bu araçlardan bir tanesidir. JSON'un resmi sitesine girdiğiniz zaman hem JSON hakkında genel bilgilere hemde JSON'ı kullanabilen dillere ve ekstra olarak JSON kullanabilmek için o dile ait `API` v.s ulaşabilirsiniz.

```
http://www.json.org/
```

JSON'un resmi adresinde `Türkçe` dil desteği de mevcuttur.

1.3 JSON Nerede Kullanılır?

Şimdi geldik en kilit soruya: JSON nerede kullanılır? Genel olarak JSON'dan bahsettik peki hakikaten JSON nerede kullanılır, şimdi bu sorunun cevabını arayalım.
JSON'u istediğiniz herhangi bir yerde kullanabilirsiniz ki daha önce JSON için programlama dillerinden bağımsızdır demiştik. Lakin genel itibari ile web ortamında sıklıkla JSON kullanılır.

Örneğin;

- Ağ üzerinden veri alışverişlerinde sıklıkla kullanılır.
- Server ve web sayfaları arasında haberleşmeyi sağlamak için kullanılır.
- En önemli nokta ise; `Web Service (Web Servis)`'ler ile kullanılır. Burada web servisler ile kullanımında devreye `Ajax` gibi bazı yapılarda işin içine girer lakin şimdilik bu kadarının bilinmesi

yeterlidir.
- Son olarak JSON, JavaScript tabanlı herhangi bir web sayfasında rahatlıkla kullanılabilmektedir.

1.4 JSON Nasıl Kullanılır?

JSON genel itibariyle iki farklı şekillerde kullanılır. Bunlardan birincisi: JSON formatında ayrı bir dosya şeklinde kullanılması, ikincisi ise web sayfalarında JavaScript etiketleri arasında kullanımıdır.

JSON formatında ayrı bir dosya şeklinde kullanılması durumunda; veriler harici bir JSON dosyasında tutulur. Bu JSON dosyasının uzantısı `.json` olmalıdır. Örneğin XML dosyasının uzantısı da `.xml` olur. Oluşturulan bu .json uzantılı dosya programlama dilleri arasında haberleşmeyi sağlar. Örneğin elimizde; `dataset.json` adında bir dosyamız olsun. Bu dosyamızı `PHP` ile açıp değiştirebilirim. Içindeki verileri okuyabilir ve yeni veriler yazabilirim. Daha sonra bu dosyayı `Java`, `Python`, `Ruby` hatta ve hatta `C` ile bile bu dosyaya erişebilirim. Burada herhangi bir sıkıntı yoktur.

Bir diğer kullanım şekli de değindiğimiz gibi web sayfalarının içerisinde JavaScript etiketleri arasında kullanımıdır. Bildiğiniz gibi bir web sayfasında, yani `HTML` kodları arasında JavaScript'i gömülü olarak kullanabiliriz. Bu kullanımda `<script></script>` etiketleri ile olur.örneğin önceki sayfalarda verdiğimiz örneğimizi `script` etiketleri arasında yazmak istersek;

```
<script type="text/javascript">
var object = {
 "uyelerTablosu": [
    {
      "uyeAd" : "Emre Can",
      "uyeSoyad" : "OZTAS"
    },
    {
      "uyeAd" : "Hazal",
      "uyeSoyad" : "ONEY"
    }
 ]
};
</script>
```

Şeklinde olacaktır.

BÖLÜM 2:
JSON İçin Araçlar

Bu bölümde genel olarak JSON için kullanılabilecek araçlardan bahsedeceğiz. Bu araçlardan bahsetmemizin nedeni ise; daha rahat bir geliştirme ortamı sağlamaktadır. Ayrıca bu araçları kullanmak zorunda değilsiniz. Herhangi bir editör yardımıyla da JSON kodlaması yapabilirsiniz. Bunun dışında JSON kodlamak için herhangi ekstra bir kurulum veya ayar yapılmasına gerek yoktur. Neden? Çünkü JSON, JavaScript tabanlıdır. Ayrıca; kullanılan her Browser (Tarayıcı), JavaScript'i tanır ve ne yapması gerektiğini bilir. Ekstra bir bilgi olarak; tarayıcılar sadece; HTML, CSS ve JavaScript'en anlar, aklınızda bulunsun.

JSON için; Windows ortamında; Notepad kullanabilirsiniz. Daha gelişmiş bir editor isterseniz; Notepad++, Sublime Text veya Geany kullanabilirsiniz. GNU / Linux ortamında: Vim, Kate, Pluma, Gedit, Bluefish veya Geany kullanabilirsiniz. Mac kullanıcıları için herhangi bir öneri de bulunamıyorum, çünkü hiç Mac'ım olmadı (bu nasıl bir çaresizliktir).

Esktra olarak editörler hakkında küçük bir eleştiride daha doğrusu bir değerlendirme de bulunmak istiyorum. Windows ortamında kullanılan Notepad++ adında bir editör bulunmaktadır. Gördüğüm kadarıyla oldukça sık kullanılan bir editör. Lakin Notepad++ yapısı itibariyle, bana göre oldukça vasat, gereksiz bir editör. Bunun yerine Geany kullanmanız oldukça yerinde bir tercih olacaktır. Geany ile Notepad++ karşılaştırılamaz bile. Bana göre Geany her halükarda Notepad++'ın bir kaç adım önündedir. Size de tavsiyem Geany'i gibi bir aracı, her türlü platformda gönül rahatlığı ile kullanabilirsiniz. Geany kullanmak istemeseniz bile tercihiniz; Sublime Text editörden yana olmalıdır.

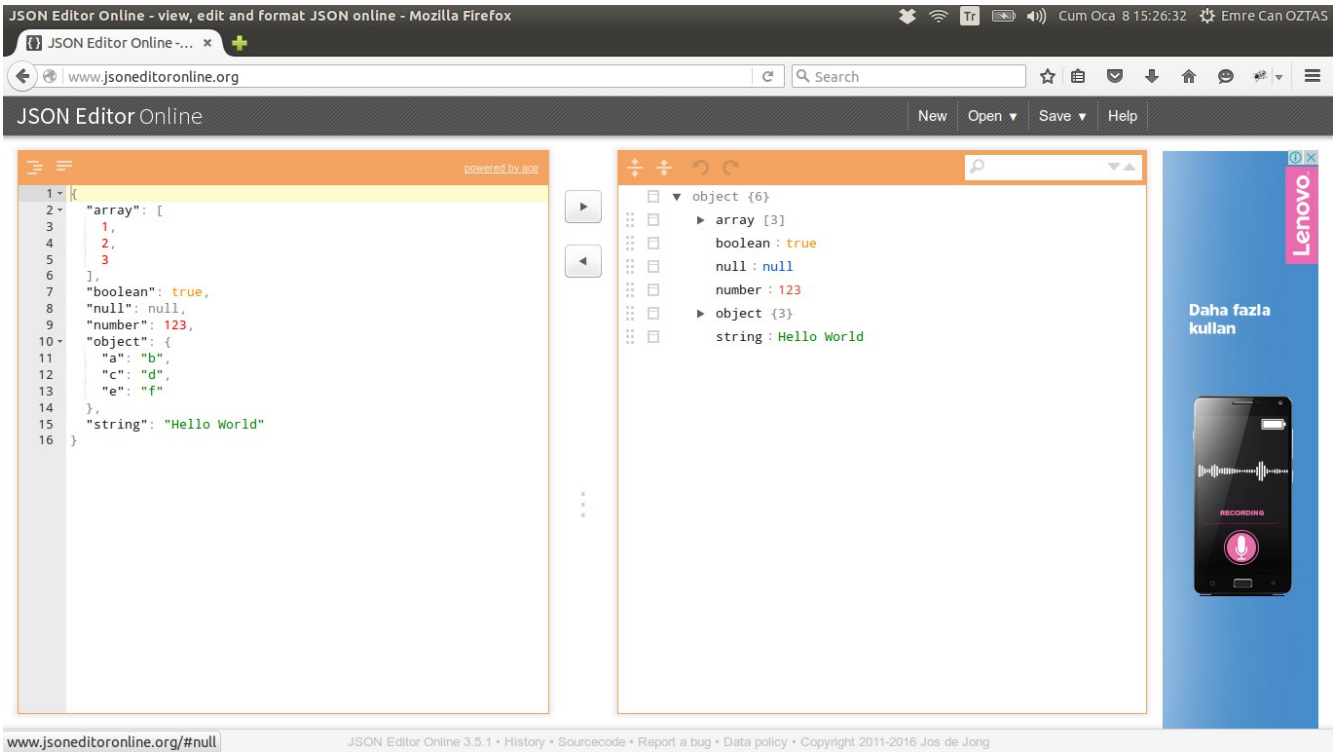
Yukarıdaki paragrafta saydığımız araçlardan herhangi birini kullanabilirsiniz. Verdiğim tavsiyeler basit işlemler içindi. Eğer herhangi bir geliştirme aracı kullanıyorsanız; Eclipse veya NetBeans, DreamWeaver, MS Visual Studio veya başka herhangi bir ortamda bu saydığım araçlar yerine bu ortamları seçmeniz en doğru çözüm olacaktır.

Ben kişisel olarak Eclipse kullananlardanım. Eclipse'nin Json Tools 1.1.0 adlı plugin'ini kurarakta JSON kodlaması yapabilirsiniz. Bunun dışında eğer NetBeans kullanıyorsanız ve NetBeans'ın HTML/JavaScript, PHP veya All in one sürümlerinden herhangi birini kullanıyorsanız, JSON için ekstra bir plugin kurmanıza gerek yoktur. Bu saydığım NetBeans sürümleri içerinde; JSON aracı kurulu olarak gelmektedir.

Herşey burada bitmedi. Yukarıdaki ortamların dışında eğer web tabanlı bir platform kullanmak isterseniz, aşağıdaki siteyi ziyaret etmenizi tavsiye ederim.

```
http://www.jsoneditoronline.org/
```

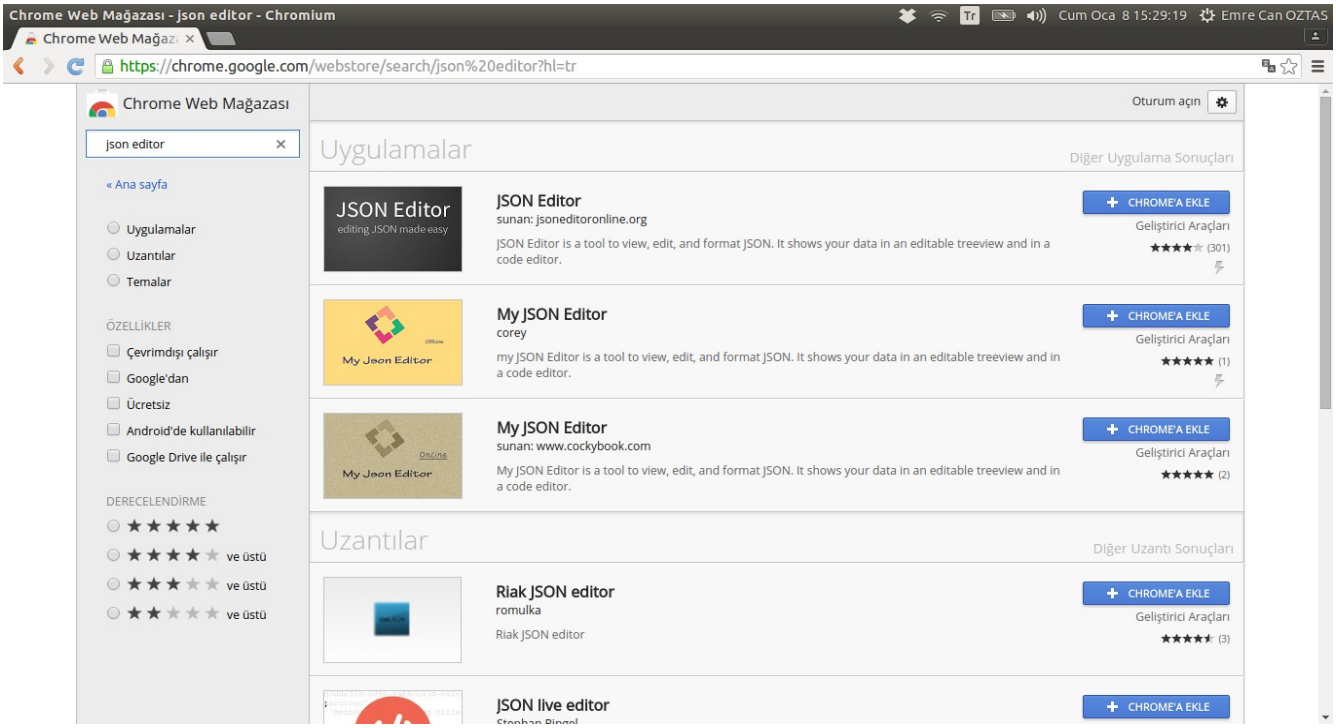
Yukarıdaki adrese gittiğiniz zaman sizi aşağıdaki sayfa karşılayacaktır.



Yukarıdaki ekranda da görüldüğü gibi iki tane alan var. Bu alanlardan sol taraftaki alana JSON kodlarınızı yazıp sağ taraftaki alanda kontrol edebilir, tanımlamış olduğunuz verilerin değerlerini ve tiplerini öğrenebilirsiniz. Aynı zamanda; renklendirme ve eş zamanlı olarak hatalarınızı gösterebilen bir yapıya sahip. Oldukça güzel ve kullanışlı olarak hazırlanmış bir sayfa, kullanmanızı tavsiye ederim. Tabiki elinin altında internet bağlantısı bulunanlara.

Eğer elinizin altında devamlı bir internet bağlantınız yoksa; yukarıdaki aracı `Chrome` eklentisi olarak edinebilirsiniz. Şuan, GNU / Linux dağıtımlarından olan `Ubuntu` kullanıyorum. Dolayısıyla sistemimde Chrome yerine `Chromium` kurulu. Chromium, Chrome'in GNU / Linux kullanıcıları için uyarlamasıdır. Eğer sizde GNU / Linux kullanıcısı iseniz Chromium kullanmanızda herhangi bir sıkıntı yoktur.

Nerde kalmıştık? Evet, Chrome veya Chromium kullanıcıları, `Chrome Web Mağazası`'na girerek; `json editor` anahtar kelimesiyle aratmaları sonucu bir takım eklentilerle karşılaşacaklardır. Örnek olması açısından aşağıdaki resmi inceleyelim.



Yukarıdaki ekran alıntısında da görüldüğü gibi ilk sırada gelen eklentiyi yükledim. Şayet kullanmak isterseniz tek tıklamayla bu eklentiyi Chrome (veya Chromium) tarayıcınıza ekleyebilirsiniz.

Yukarıdaki bahsetmiş olduğum eklentiyi, Chrome (veya Chromium) tarayıcınıza eklenti olarak eklediğinizde ilk resimde gösterilen ekranla karşılaşacaksınız.

Belirli araçlardan veya editorlerden bahsettik bunlar arasında seçim yapmak size kalmış. Lakin daha öncede belirttiğim gibi herhangi bir proje üzerinde çalışıyorsanız veya gelişmiş araçlarınız varsa bunları kullanmanız en mantıklı seçim olacaktır. Örneğin Eclipse üzerinde `Java` kodluyorsanız kalkıpta ekstra bir editor veya araç aramanız gereksizdir. O araç üzerinde devam etmeniz en mantıklı olanıdır.

BÖLÜM 3:
Syntax

Bu bölümde temel olarak `JSON Syntax'ı (Söz Dizimi)` üzerinde duracağız. Temel olarak JSON, zaten bir gösterim şeklidir. `XML`'in nasıl ki bir yapısı varsa JSON'un da buna benzer bir yapısı vardır. Konumuza başlamadan önce yine tekrar edelim: JSON bir programlama dili değildir! JSON bir veri taşıma formatıdır!

Ilk olarak basit örnekler üzerinde duralım ve JSON'u temelden öğrenmeye çalışalım.

JSON'da veriler: `name/value` şeklinde bulunur. Yani bir veri olacak ve bu verinin de bir ismi olacaktır. Bu veriyi de ismine göre çağıracağız. Bu yapıyı temel olarak bir değişken ve değeri olarak düşünebilirsiniz.

Aşağıdaki örneğimizi inceleyelim.

```
"adSoyad": "emre can oztas"
```

Yukarıdaki örneğimizde; `adSoyad` adında bir veri alanı var. Bu veri alanında da bir değer var. Burada dikkat edilmesi gereken nokta; `name` yani veri alanı çift tırnak (“ ”) içerisinde yazılmasıdır. Bu hep böyledir. name alanları çift tırnak işaretleri arasına yazılmalıdır. Bu konu üzerinde ilerleyen bölümlerde detaylı olarak duracağız. name alanları çift tırnak içerisinde `string (metinsel)` şekilde yazılırken; veri kısmında bu biraz farklıdır. Çünkü verinin tipi değişebilir. Bu veri tipi; string bir yapıda olabileceği gibi `integer (tam sayı)`, `float (kayan noktalı sayı)` veya diğer herhangi bir veri tipinde olabilir. Şuan için bunun üzerinde fazla durmayalım. Bir diğer bölümde JSON'da bulunan veri tiplerini detaylı olarak inceleyeceğiz. Şimdilik sadece bu kadarını bilmemiz yeterli.

Yukarıdaki örneğimize yeni veriler de ekleyebiliriz. Aşağıdaki örneğimizi inceleyelim.

```
"adSoyad": "emre can oztas", "meslek": "bilgisayar muhendisi"
```

Yukarıdaki örneğimize yeni bir veri alanı daha ekledik. Bu iki alanı da birbirinden ayırmak içinde virgül (,) işaretini kullandık. Buradan çıkacak sonuç: tek bir veri olabileceği gibi yan yana sıralı veriler de olabilir. O zamanda verileri birbirlerinden ayırmak için virgül işaretini kullanmamız gerekir. Bunu unutmayalım.

Peki. Devam edelim. Yukarıdaki alanlarımıza string (metinsel) değerler yerine ekstra olarak sayısal bir değer daha ekleyelim. Örneğimiz aşağıdaki gibi olacaktır.

```
{"adSoyad": "emre can oztas", "meslek": "bilgisayar muhendisi", "yil":
2015}
```

Yukarıdaki örneğimizi inceleyelim. Sanırım işler biraz daha karıştı. Ama panik yok. Adım adım ilerliyoruz. Sıralı verilerin arasına virgül işaretinin konacağından bahsetmiştik. Veri satırımıza; `yil` adında yeni bir veri alanı daha ekledik. Bu veri alanı sayısal bir alan olduğu için veri kısmında çift tırnak kullanmadık. Buraya kadar bir sıkıntı yok ama küme parantezleri (süslü parantez, güzel parantez v.s nasıl adlandırırsanız) de olaya

dahil oldu. Hemen açıklık getirelim. JSON verileri, küme parantezleri { } arasında yazılmalıdır. Yani eğer JSON formatında bir veri kümesi hazırlıyorsanız; bu veri kümesi küme parantezleri arasında yazılmalıdır. Bu bir kuraldır.

Yukarıdaki son yazmış olduğumuz örneğimizi daha okunaklı bir hale getirelim ve JSON'un “Okunması Kolaydır!” sözünü yerine getirmiş olalım. Bundan sonraki örneklerimizi de aynı format ile yazmaya çalışacağız. Sizde böyle kullanmaya çalışırsanız, karışıklığa sebebiyet vermemiş olursunuz.

Örneğimizin yazım şekli aşağıdaki gibi olacaktır.

```
{
    "adSoyad": "emre can oztas",
    "meslek": "bilgisayar muhendisi",
    "yil": 2015
}
```

Sanırım böyle yazmamız doğru bir tercih oldu.

3.1 JSON Array

Bu başlık altında, birden fazla satırda; JSON formatında veriler nasıl yazılır bunu incelemeye çalışacağız. Ilk olarak yukarıda yazmış olduğumuz örneğimizi alalım ve yeni veri satırları ekleyelim yani örneğimizi biraz daha genişletelim. Örneğimizin son şekli aşağıdaki gibi olacaktır. Aşağıdaki örneğimizi inceleyelim.

```
{
 [
     {
      "adSoyad": "emre can oztas",
      "meslek": "bilgisayar muhendisi",
      "yil": 2015
     },
     {
      "adSoyad": "rozerin aktas",
      "meslek": "bilgisayar muhendisi",
      "yil": 2014
     },
     {
      "adSoyad": "muhammet kucukbardasli",
      "meslek": "doktor",
      "yil": 2013
     }
 ]
}
```

Yukarıdaki örneğimizi inceleyelim. Ilk örneğimizi aldık ve yeni veri satırları ekledik. Lakin burda da işin içine köşeli parantezler [. . .] girdi. Köşeli parantezler, çoğu programlama dilinde, diziler için kullanılan karakterlerdir. JSON'da da durum farklı değildir.

Örneğimizi satır satır açıklamaya çalışalım ve JSON, array (dizi) yapısını anlamaya çalışalım. Öncelikle; JSON satırlarının küme parantezleri arasında yazılması gerektiğinden bahsetmiştik. Burada herhangi bir

sıkıntı yok. Daha sonra köşeli parantezlerimizi [ ] açtık ve yazacağımız satırların bir dizinin elemanları olacağını belirttik. Daha sonra küme parantezlerimizi açtık. Burada küme parantezleri; oluşturulan dizinin her bir satırını belirtiyor. Yani bir dizimiz var ve bu dizimizinde elemanları var. ne demek istediğimi aşağıdaki yapımızı incelediğimizde daha iyi anlayacağımızı sanıyorum.

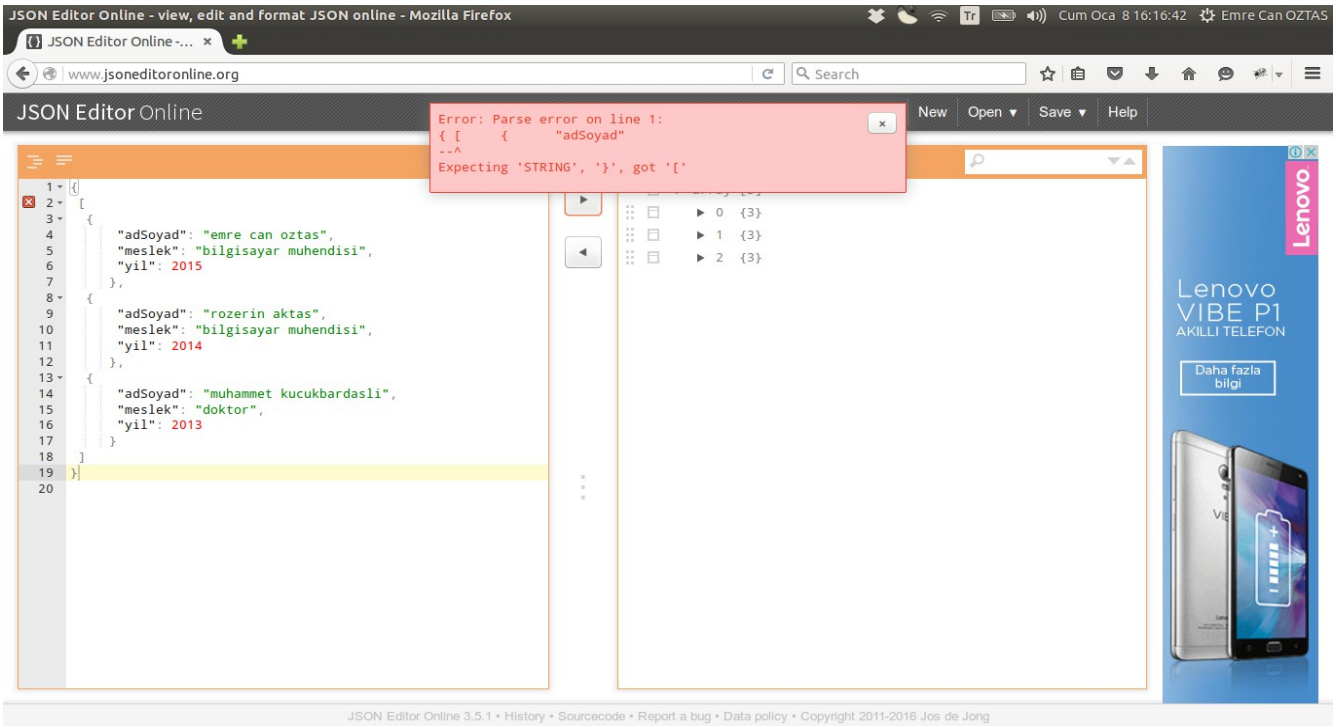
adSoyad	meslek	yil
emre can oztas	bilgisayar muhendisi	2015
rozerin aktas	bilgisayar muhendisi	2014
muhammet kucukbardasli	doktor	2013

Yukarıdaki tablomuz, yazmış olduğumuz dizimizin gösterim şeklidir. Yani yukarıdaki tablomuzu dizi şeklinde JSON veri satırı olarak yazmış olduk.

Tamam buraya kadar bir sıkıntımız yok. Şimdi bir de JSON formatındaki verilerimizi editor ortamında görelim bakalım herhangi bir hatamız var mı? Hatamız yoksa da ekran çıktısına bakalım ve ifade etmek istediğimizi tam karşılıyor mu görelim.

Ikinci bölümde bahsetmiş olduğumuz editorü açalım ve veri satırlarımızı yazalım.

Örnek ekran çıktımız aşağıdaki gibi olacaktır.

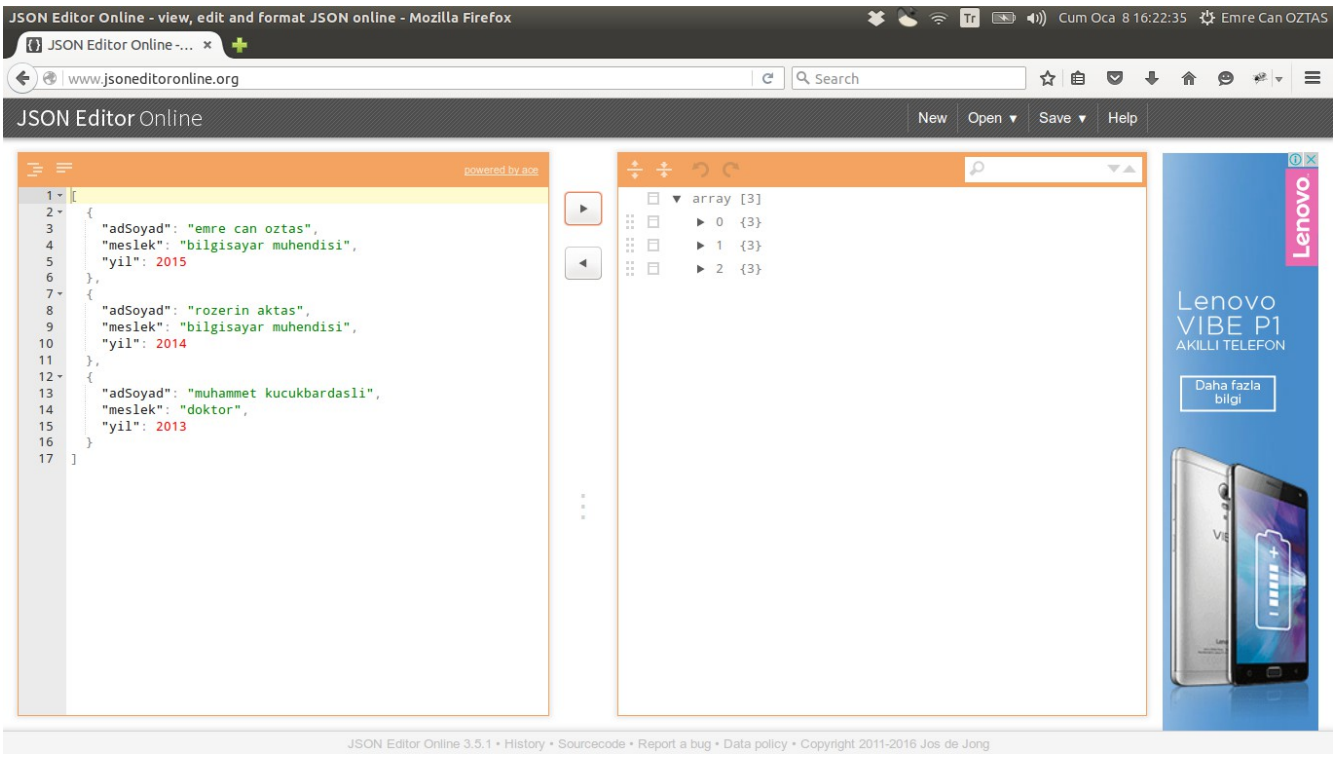


Yukarıdaki ekran çıktısında da görüldüğü gibi; hata aldık! Bu hatamızın nedeni nedir? Herhangi bir tahmininiz yoksa cevap vereyim. Tamam dizimizi tanımladık lakin bu dizimize bir isim vermedik. Değil mi? Iste hatamızın nedeni burada.

Şayet yukarıdaki örneğimizde; en başta küme parantezleri içerisinde dizimizi yazmamış olsa idik, yani:

```
[
     {
      "adSoyad": "emre can oztas",
      "meslek": "bilgisayar muhendisi",
      "yil": 2015
     },
     {
      "adSoyad": "rozerin aktas",
      "meslek": "bilgisayar muhendisi",
      "yil": 2014
     },
     {
      "adSoyad": "muhammet kucukbardasli",
      "meslek": "doktor",
      "yil": 2013
     }
]
```

Şeklinde olsaydı, o zaman herhangi bir hata mesajıyla karşılaşmayacaktık. Neden? Çünkü JSON editör bu yapıyı sadece dizi syntax (söz dizimi) bakımından kontrol edecek ve herhangi bir hata mesajı vermeyecekti. Isterseniz deneyim görelim.



Görüldüğü gibi herhangi bir hata yok. Çünkü bu yapı sadece bir dizi herhangi bir JSON yapısında değil. Ayrıca editörün sağ tarafına bakarsanız yazmış olduğumuz satırlarımızın bir array (dizi) olduğunu bize göstermekte.

Peki ne yapmamız lazım? JSON'da bir diziyi nasıl isimlendirme yapmak lazım? Hemen buna bakalım şimdi.

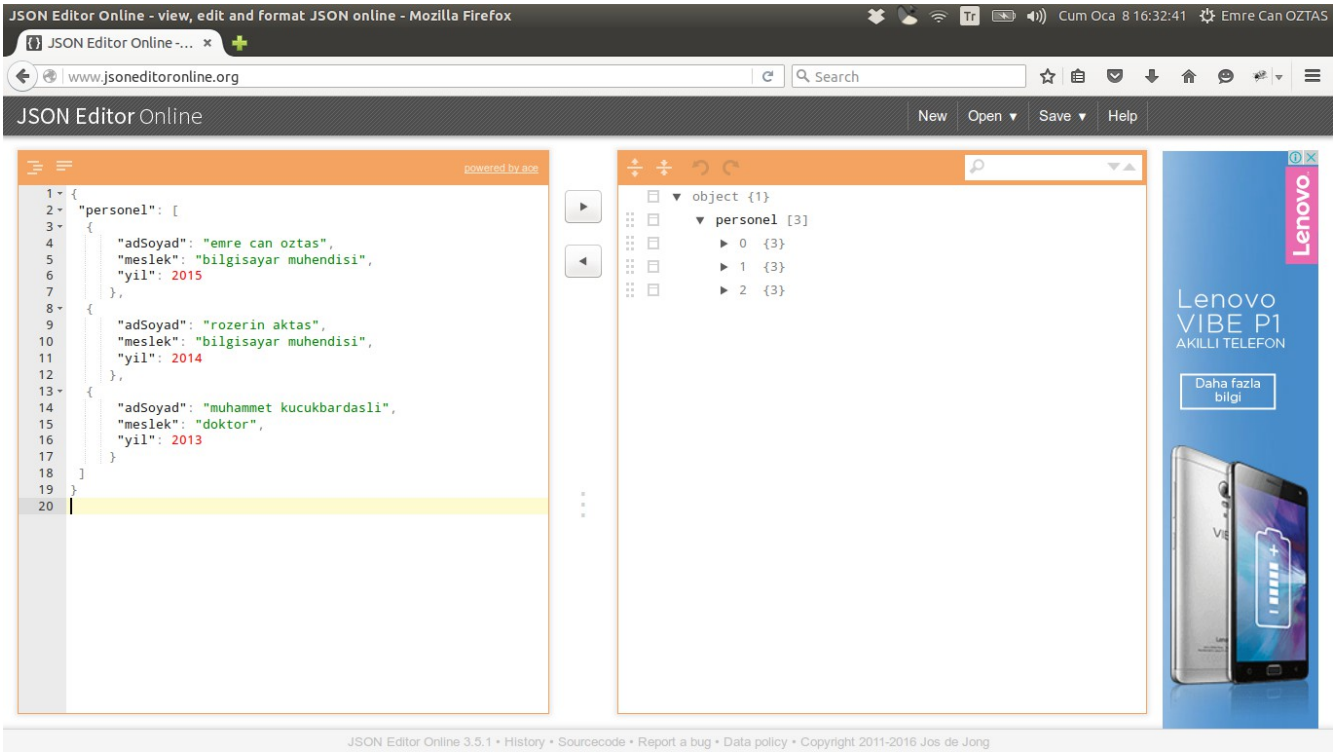
JSON'da diziyi isimlendirmek, tıpkı name / value gibi. Yine name (isim) alanını çift tırnak içerisinde

yazmamız gerekiyor. Daha iyi açıklama yapmak için yukarıdaki örneğimize bir isim verelim. Örneğimizin son şekli aşağıdaki gibi olacaktır.

```
{
 “personel”: [
     {
      "adSoyad": "emre can oztas",
      "meslek": "bilgisayar muhendisi",
      "yil": 2015
     },
     {
      "adSoyad": "rozerin aktas",
      "meslek": "bilgisayar muhendisi",
      "yil": 2014
     },
     {
      "adSoyad": "muhammet kucukbardasli",
      "meslek": "doktor",
      "yil": 2013
     }
 ]
}
```

Yukarıdaki örneğimizde görüldüğü gibi dizimize `personel` adını verdik. Burada da dikkat edilmesi gereken nokta tıpkı name alanında olduğu gibi isimlendirmemiz çift tırnak içerisinde olmalı. Hadd-i zatında normal bir veri satırı ile dizi arasında pek bir fark yok gördüğünüz gibi.

Örneğimizi editör ortamında deneyelim. Bakalım bu sefer hata mesajıyla karşılacak mıyız?



Görüldüğü gibi herhangi bir hata mesajıyla karşılaşmadık.

Diziler konusuyla örneğimizi yaptık. Peki diziler hep aynı yapıda mı olmalıdır? Yani demek istediğim; dizi içerisinde hep aynı alanlar mı olmalıdır? Hayır öyle bir zorunluluk yoktur. Dizi içerisindeki alanlar değişebilir. Yukarıdaki örneğimiz üzerinde biraz oynama yapalım. Örneğimizin son şekli aşağıdaki gibi olacaktır.

```
{
 "personel": [
     {
      "adSoyad": "emre can oztas",
      "meslek": "bilgisayar muhendisi"
     },
     {
      "adSoyad": "rozerin aktas"
     },
     {
      "adSoyad": "muhammet kucukbardasli",
      "yil": 2013
     }
 ]
}
```

Yukarıdaki örneğimizde; dizinin ilk satırından: `yil`, ikinci satırından: `meslek` ve `yil`, üçüncü satırından da: `meslek` alanlarını sildim. Dizimiz farklı bir hal aldı değil mi? Peki böyle kullanım olabilir mi? Elbette olabilir. Istediğiniz şekilde kullanabilirsiniz. Tamamen size kalmış durumda. Geçelim bir diğer konuya. Dizi içerisine yeni bir tanımlama yapılabilinir mi? Bu sorusunun cevabı da evet. Dizi içerisine yeni bir tanımlama yapılabilir. Lakin burada sadece name alanı bulunmak zorundadır. Yani value alanı olmaz. Bu kavramı da açıklamak için dizimize yeni bir tanımlama daha ekleyelim. Örneğimizin son şekli aşağıdaki gibi olacaktır.

```
{
 "personel": [
     {
      "adSoyad": "emre can oztas",
      "meslek": "bilgisayar muhendisi"
     },
     {
      "adSoyad": "rozerin aktas"
     },
     {
      "adSoyad": "muhammet kucukbardasli",
      "yil": 2013
     },
      "Benim dizim"
 ]
}
```

Yukarıdaki örneğimizde; dizimiz içerisine name alanı ekledik ve bu alana “`Benim dizim`” yazdık. Aslında burada olan nedir biliyor musunuz? Benim dizim olarak yazdığımız alan da dizimizin bir elemanı gibi algılanır. Yani dizimizde artık 4 tane veri satırı var. Buradaki kavram bizi başka bir yere götürecektir. O da nedir biliyor musunuz? Köşeli parantezler kullanmak yerine sadece value alanına sahip olan dizi

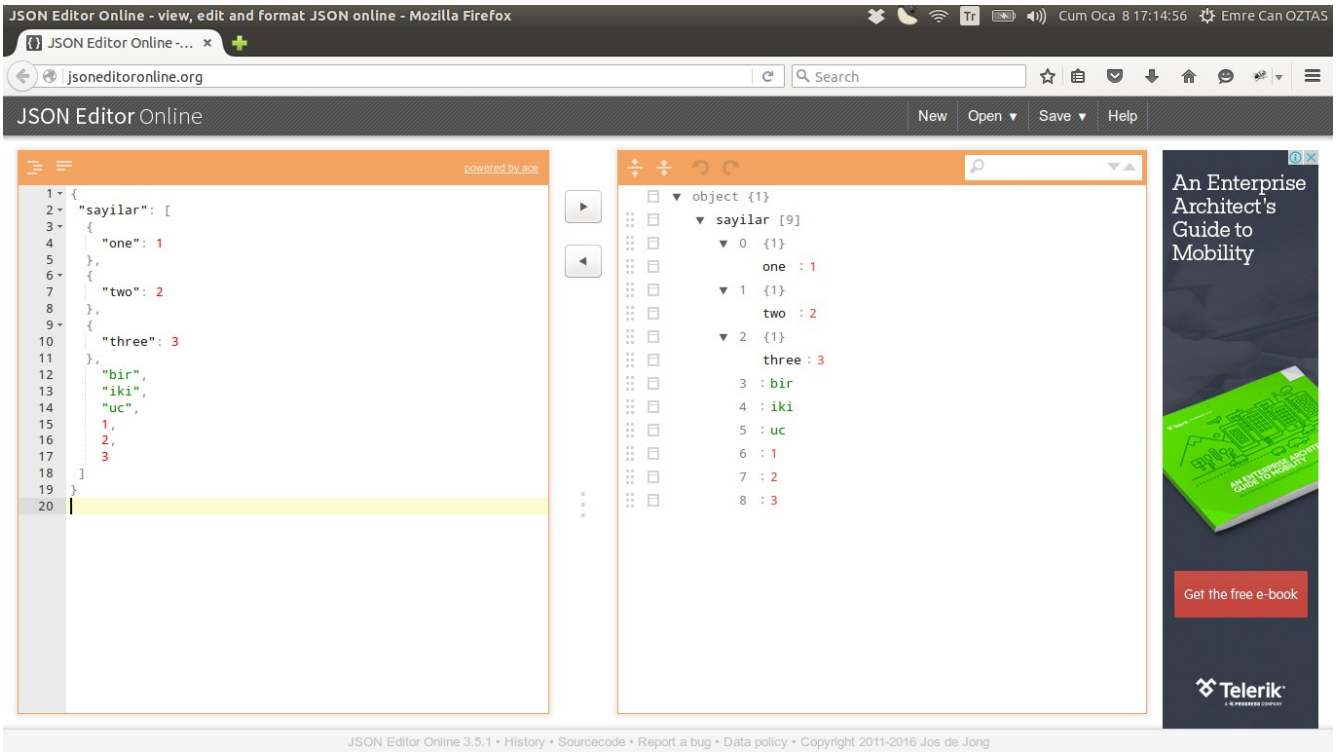
elemanları yazılabilir. Bununla ilgili aşağıdaki örneğimizi inceleyelim.

```
{
 "sayilar": [
    1,
    2,
    3,
    4,
    5
 ]
}
```

Yukarıda örneğimizde sadece dizinin içeriğini yazdık. Herhangi bir name alanına sahip değil. Bu şekilde de kullanılabilir. Lakin burada dikkat edilmesi gereken en önemli nokta: dizi içerisinde name / value şeklinde bir yapı kurulacaksa o zaman küme parantezleri kullanılmalıdır. Veyahut name / value değil de sadece value şeklinde yazılacaksa o zaman küme parantezlerini kullanmamıza gerek yoktur. Açıklamasını yaptığımız bu kavramları bir örnek üzerinde gösterelim. Örneğimiz aşağıdaki gibi olacaktır.

```
{
 "sayilar": [
  {
    "one": 1
  },
  {
    "two": 2
  },
  {
    "three": 3
  },
    "bir",
    "iki",
    "uc",
    1,
    2,
    3
 ]
}
```

Yukarıdaki örneğimize ait olan ekran çıktısını alalım.



Yukarıdaki ekran çıktısında da görüldüğü gibi; diziler, küme parantezleri arasında: name / value, küme parantezleri dışında sadece value değeri olan elemanlara sahip olabilirler.

Bir JSON dosyasında ya da yapısında birden fazla dizi olabilir. Dizilerle, dizi olmayan satırlar bir arada bulunabilir. Bununla ilgili bir örnek gerçekleştireceğiz. Ama burada değinmek istediğim son bir konu var. Dizi içerisinde diziler yazabilirsiniz. Burada herhangi bir sıkıntı yoktur. Istenildiği kadar iç içe diziler oluşturulabilir ve kullanılabilir. Örneğin;

```
{
  "diziOne": [
    {
      "diziTwo": [
        {

        }
      ]
    }
  ]
}
```

Iç içe 2 dizi yazdık. Herhangi bir problem yok. Içteki diziye bir dizi daha ekleyelim.

```
{
  "diziOne": [
    {
      "diziTwo": [
        {
          "diziThree": [
```

```
            {

             }
          ]
        }
      ]
    }
  ]
}
```

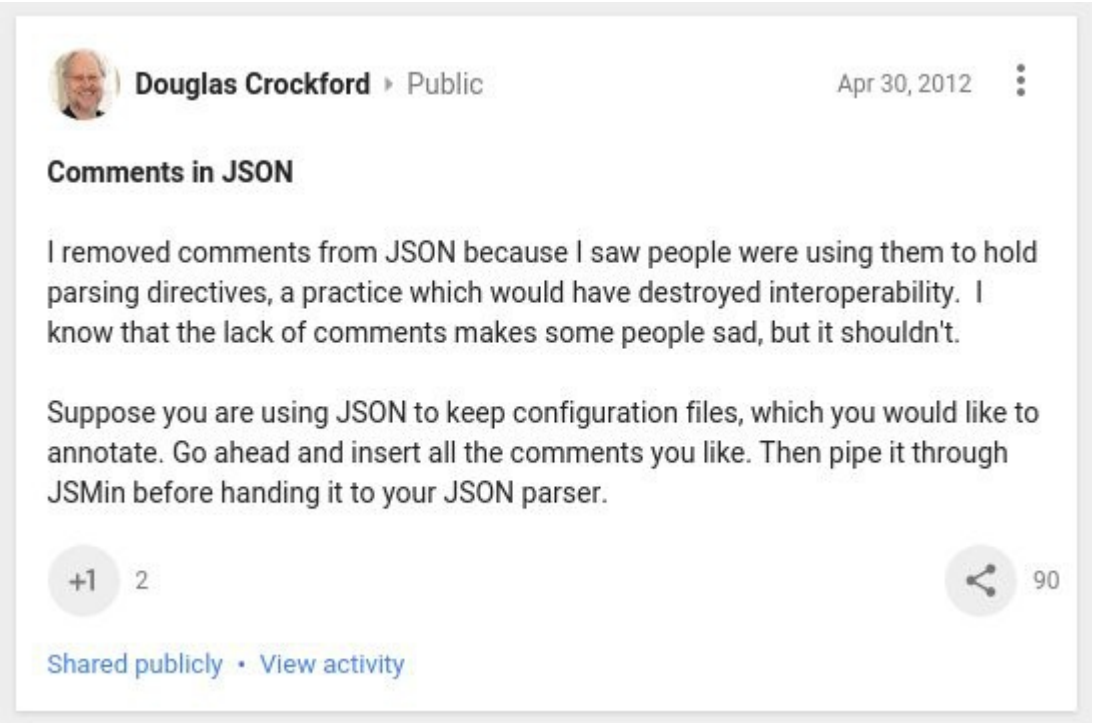
Görüldüğü gibi iç içe diziler alıp başını gidiyor. Bu kullanım yöntemlerini göstermek bizden kullanmak sizden diyelim ve diziler konusunu burada sonlandıralım.

3.2 Comment Line

Buraya kadar olan bölümde; `Comment Lines (Açıklama Satırları)` hakkında hiç bahsetmedik. Çünkü JSON'da açıklama satırları bulunmaz. Aslında vardı yani varmış bir zamanlar (ben açıklama satırları olan devire yetişemedim). JSON'un ilk zamanları aşağıdaki şekillerde açıklama satırları eklenebiliyordu.

```
{
    # this is a comment line
    // this is an another comment line
    "name": "emre can",
    "surname": "oztas",
    "country": "turkey",
    "city": "ankara"
}
```

Artık JSON yapısında, herhangi bir açıklama satırı kullanılamıyor. JSON'ı geliştiren; `Douglas Crockford`, `Google+` hesabından aşağıdaki açıklamayı yaparak, JSON'daki açıklama satırlarını kaldırdığını ve nedenlerini açıklamıştır.



Açıklama satırları yazmak istiyorsanız; name/value kuralına göre yazabilirsiniz. Aşağıdaki örneğimizi inceleyelim.

```
{
    "_comment": "this is a comment line",
    "name": "emre can",
    "surname": "oztas",
    "country": "turkey",
    "city": "ankara"
}
```

Yukarıdaki örneğimizde görüldüğü gibi; `_comment`, name alanı açtık ve açıklama satırımızı yazdık. Burada da dikkat edilmesi gereken nokta; işiniz bittiği anda yani, yaptığınız işi tamamladığınızda açmış olduğunuz name/value alanını silmek. Neden? Çünkü bu açmış olduğunuz alan bir veri satırı gibi algılanıyor ve ona göre yapılacak işlemlere dahil olabilir. O yüzden buna çok dikkat edelim.

3.3 Extras

Bu bölümde ekstra olarak değinmediğimiz veya satır aralarında kalan konuları ele alacağız. Bu konuları ele almamızın nedeni; daha önce aklınıza takılmış veya sonradan farkına varabileceğiniz konulara açıklık getirmektir.

Öncelikle JSON dosyaları diske: .json uzantısıyla kaydedilir, bunu biliyoruz. Bu kaydedilen dosya herhangi bir programlama dili yardımıyla açılıp okunabilir veya düzenlenebilir, bunu da biliyoruz. Lakin .json uzantısıyla diske kaydedilen JSON dosyası, JavaScript ile açıp düzenlenemez. Neden? Çünkü JavaScript'in böyle bir özelliği yoktur. Buna dikkat edelim.

Bir diğer konuda; dikkat ederseniz, herhangi bir noktalı virgül `(;)` kullanmadık. JSON, yapısı itibari ile noktalı virgülü desteklemez. O yüzden diğer programlama dilllerinde olan; satır sonlarına noktalı virgül konulması alışkanlığı JSON'da hata almanıza sebep olacaktır.

Yazdığımız örneklerimizde; `Türkçe` karakter kullanmadık. Dikkat etmiş olabilirsiniz. Burada Türkçe karakter kullanmamızın tek sebebi benim. `Ingilizce` yazılım kuralına göre örneklerimizi yazdık lakin Türkçe karakterlerde kullanabilirsiniz. `name` alanında kullanmamaya özen gösterelim fakat `value` alanında istediğimiz gibi Türkçe karakter kullanabilirsiniz. Aşağıdaki örneğimizde olduğu gibi.

```
{
    "name": "Emre Can",
    "surname": "ÖZTAŞ",
    "country": "Türkiye",
    "city": "Ankara"
}
```

BÖLÜM 4:
Data Types

Bu bölümde; JSON'ın desteklediği `Data Types (Veri Tipleri)` üzerinde duracağız. Buraya kadar olan bölümde; `string` ve `integer` sayılar üzerinden örnek verdik. Bunlar dışında JSON'da 4 veri tipi daha vardır. Yani toplamda `6` veri tipine sahiptir. Gelin isterseniz bu veri tipleri neler bunları bir tablo halinde görmeye çalışalım daha sonra da detaylı olarak ayrı başlıklar altında incelemeyelim.

Aşağıdaki tablomuza bakalım.

JSON Data Types
String
Number
Array
Object
Boolean
Null

Şimdi yukarıdaki tablomuzda veri tiplerini ayrı ayrı başlıklar altında incelemeye çalışalım.

4.1 String

`string`, bildiğiniz gibi metinsel ifadelerde kullanılan bir tiptir ve her programlama dilinde vardır. Önceki bölümlerde string kullanarak örneklerimizi gerçekleştirdik ve `name` alanının her zaman string olması gerektiğinden bahsettik. string veri tipini anlamak için basit bir örnek yazalım. Örneğimiz aşağıdaki gibi olacaktır.

```
{
    "filmAdi": "Fight Club",
    "yonetmen": "Chuck Palahniuk",
    "basroller": "Brad Pitt, Edward Norton, Helena Bonham Carter"
}
```

Yukarıdaki örneğimizde; value alanlarının hepsi string bir yapıda. Bunu daha önce gördük. Burada dikkat edilmesi gereken nokta; value alanlarında istenilen kadar boşluk bırakılabilir. Tek şart: string ifadelerin çift tırnak `(“”)` içerisinde yazılması gerektiğidir. Bir diğer noktada; `basroller`, name alanı dizi şeklinde yazılabilirdi. Önceki bölümlerden zaten biliyoruz fakat onu da `Array` başlığı altında görelim.

4.1.1 Escape Case

Her programlama dilinde olduğu gibi JSON'da da `Escape Case (Kaçış Karakterleri)` vardır. Kaçış karakterlerini bilmeyen veya unutanlar için hatırlatma yapalım. String ifadeler içerisinde; belli

karakterler kullanılamaz. Örneğin; çift tırnak kullanılamaz. Kaçış karakterleri böyle durumlarda imdadımıza koşar. Peki kaçış karakterleri nelerdir? Aşağıdaki tablomuza bakalım.

“	double quotation
\	reverse solidus
/	Solidus
b	Backspace
f	form feed
n	new line
r	carriage return
t	horizontal tab
u	unicode

JSON yukarıdaki kaçış karakterinin hepsini desteklemektedir. Herhangi bir hata vermemektedir. Lakin şöyle de bir durum var: yukarıdaki kaçış karakterlerini yazdığınızda ekranda çıktısı olarak herhangi bir değişiklik olmamaktadır. Örneğin; \” eklediğiniz zaman string içerisine tırnak işaretini eklemektedir ama \ işareti de string ifade içerisinde kalmaktadır. O yüzden kullanımı pek tavsiye edilmemektedir. Ilerleyen zamanlarda, yorum satırları gibi bu ifadelerde JSON'dan kaldırılırsa pek şaşırmayın.

Yukarıdaki kaçış karakterlerinden, bana göre en kullanışlı olanı: u karakteri. Bildiğiniz gibi belli bazı karakterleri kullanmak için unicode karakterler kullanmak gerekir. Peki unicode nedir? Unicode; dünya üzerinde varolan bir çok karakter, sembol veya ifadeleri kullanmak için oluşturulmuş ve belli numaralar verilerek biraraya getirilmiş bir karakterler kümesidir. Detaylı bilgi için aşağıdaki adrese bakabilirsiniz.

```
http://unicode.org/
```

Oluşturulan bu kümedeki karakteri kullanmak için de aşağıdaki adresi ziyaret edebilirsiniz.

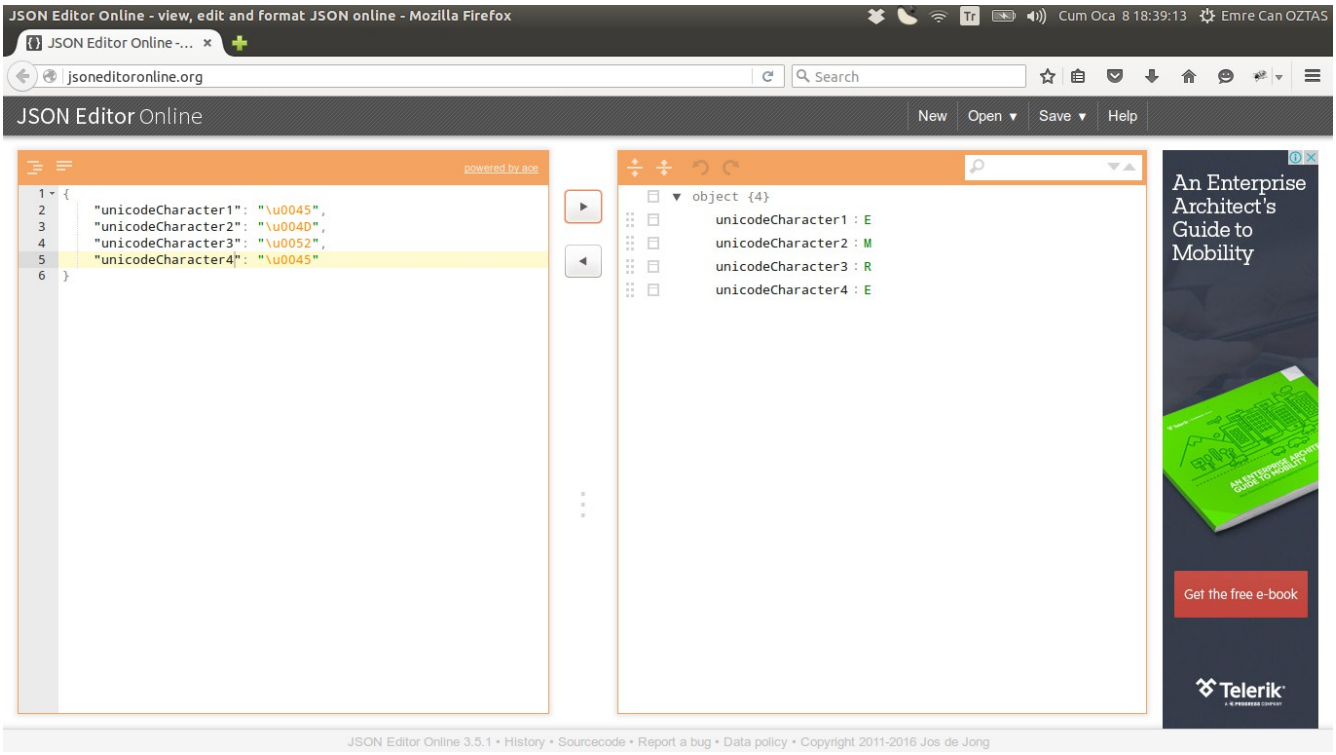
```
http://unicode-table.com/en/
```

Unicode karakterler: 4 basamaklı hexadecimal (onaltılık) sayılardan oluşmaktadır. Bir örnek üzerinden açıklamaya çalışalım. Örneğimiz aşağıdaki gibi olacaktır.

```
{
    "unicodeCharacter1": "\u0045",
    "unicodeCharacter2": "\u004D",
    "unicodeCharacter3": "\u0052",
    "unicodeCharacter4": "\u0045"
}
```

Yukarıdaki örneğimizde de görüldüğü gibi: \u ifadesinden sonra 4 basamaklı hexadecimal bir sayı yazarak istediğimiz bir karakteri kullanabiliyoruz.

Yukarıdaki örneğimizin ekran çıktısını alacak olursak;



şeklinde olacaktır.

Ikinci vermiş olduğum adrese bakarsanız, daha pek çok karakter demetine ulaşabilirsiniz. Aralarından seçtiklerinizi unicode formatı şeklinde kullanabilirsiniz.

4.2 Number

`Number (Sayı)` hayatımızın bir parçası. Onlar olmadan hayat çok sözel olurdu herhalde. JSON'da number olarak nitelendirilen value alanları: integer (tam sayı) ve float (kayan noktalı) değerler gelebilir. Kayan noktalı alanlar; JavaScript'teki yapıyla aynıdır. `Double - Precision (Çift Hassasiyetli)` sayılar gelebilir. Burada float ifadesini kullanmam tamamen farazi. Çift hassasiyetli sayılar `double` mantığındadır. Yani float tipinin saklayabileceği kapasitesinin iki katı kadardır. Bir örnek üzerinde ne demek istediğimizi açıklamaya çalışalım. Örneğimiz aşağıdaki gibi olacaktır.

```
{
    "numberOne": 639,
    "numberTwo": 3.141818
}
```

Yukarıdaki örneğimizde hem tam sayı tipinde hemde kayan noktalı sayı tipinde değerler tanımladık. Yani burada; tam sayı veya kayan noktalı sayı diye bir ayrım söz konusu değildir. Buna dikkat edelim.

4.3 Array

Array (Dizi), konusu bir önceki bölümde görmüştük, lakin sadece benzer yapıdaki değerlerin kümelenmesi olarak bahsetmiştik. Burada da durum farklı değil basit bir örnek yaparak diğer bir veri tipine geçelim. String kısmında yapmış olduğumuz örneğimizi tekrar alalım ve gerekli alanları dizi şeklinde yeniden oluşturalım. Örneğimiz aşağıdaki gibi olacaktır.

```
{
    "filmAdi": "Fight Club",
    "yonetmen": "Chuck Palahniuk",
    "basroller": [
          "Brad Pitt",
          "Edward Norton",
          "Helena Bonham Carter"
    ]
}
```

Yukarıdaki örneğimiz basit bir örnekti. Bir önceki bölümde dizi yapısını görmüştük zaten. Ayrıca dizi içinde dizinin de olabileceğini bir önceki bölümde görmüştük, hatırlarsanız.

4.4 Object

`Object (Nesne)` yapısı, JSON'da biraz farklıdır. Yine aynı şekilde; name / value şeklinde yazılır ama ekstra olarak bir de küme parantezleri açmamız gerekir. Ne demek istediğimi aşağıdaki örneğimizi incelediğimizde anlayacağımızı sanıyorum.

```
{
  "jsonObj":{
    "objOne": "One",
    "objTwo": "Two",
    "objThre": "Three"
  }
}
```

Yukarıdaki örneğimizde de görüldüğü gibi; jsonObj adında bir name alanı yazdık. Daha sonra bu name alanına; küme parantezleri açarak yeni satırlar ekledik. Işte bu JSON ortamında bir Object yani nesnedir. Nesnelerin kullanımı yukarıdaki gibi olmakla birlikte; JSON nesnesi içerisinde dizi de kullanılabilir, diğer herhangi bir veri tipi de kullanılabilir.

4.5 Boolean

Bildiğiniz gibi `boolean` ifadeler iki tiptir. Bunlar: `true` ve `false`. Bu ifadeler doğrudan JSON veri satırlarında kullanılabilir. Bir örnek üzerinde görmeye çalışalım. Örneğimiz aşağıdaki gibi olacaktır.

```
{
    "isEmpty": false,
    "isNumber": false,
    "isFloat": false,
    "isString": true
}
```

Yukarıdaki örneğimizde de görüldüğü gibi true / false ifadelerini bir tırnak veya herhangi bir işaret olmadan doğrudan kullanabiliriz.

4.6 Null

`Null`, bir değer ifade etmez. Örneğin bir değişken tanımladığınız zaman ilk değerini atamasanız bile o değişkenin `default (varsayılan)` bir değeri vardır. Bu da programlama dilinden diline değişmektedir. Null, 0 veya boş demek değildir, kesinlikte. Null, `hiçlik-yokluk` anlamına gelmektedir. Sanki uzaydaki bir boşluk gibi. Işte null ifadesi de, JSON'da doğrudan kullanılabilir. Nasıl mı? Aşağıdaki örneğimizi inceleyelim.

```
{
        "isString": null,
        "isNumber": null
}
```

Yukarıda örneğimizde de görüldüğü üzere, nesnelerimizin herhangi bir değeri yok. O yüzden null değerlerini atadık. Dediğimiz gibi null ifadesi doğrudan kullanılabilir. Yalnız aşağıdaki gibi bir yol izlerseniz:

```
{
    "value": "null"
}
```

Bu ifade null değildir. Value nesnesinin değeri null idir. Yani string tipindedir. Buna dikkat edelim.

Bu bölümde kısaca JSON değişken tipleri üzerinde durduk. Bundan sonraki bölümlerde; bu veri tiplerini kullanarak daha efektif JSON veri kümeleri oluşturmaya çalışacağız.

BÖLÜM 5:
JavaScript & JSON

Bu bölümde, JavaScript kodları içerisinde JSON kullanımını görmeye çalışacağız. Bildiğiniz gibi JSON, JavaScript'ten türetilmiş bir yapıdır. Dolayısıyla herhangi ekstra bir araç kullanmamıza gerek yok. Doğrudan <script></script> etiketleri arasında veya ekstra JavaScript dosyalarında kullanabiliriz.

Herşeyden önce JavaScript bilmeyenler için, JavaScript ortamında bir nesne nasıl tanımlanır buna bakalım. JavaScript, nesne tabanlı bir script dildir ve oldukça geniştir. JavaScript bir programlama dili sayılmasa bile başlı başına bir deryadır. Bu deryada kimi kaptan olur kimisi de benim gibi bir kayıkçı. Ben işime yarayan kadar JavaScript biliyorum. Eğer hiç JavaScript bilmiyorsanız, daha önce yazmış olduğum, Simple JavaScript kitabımdan faydalanabilirsiniz.

Konumuza geri dönecek olursak; JavaScript ortamında bir nesne aşağıdaki gibi oluşturulur.

```
var objectName = new ObjectType();
```

JavaScript ortamında, boş bir nesne ya da işaret edeceği tarafı boş bırakılan bir nesne de aşağıdaki şekilde oluşturulabilir.

```
var objectName = {};
```

Gördüğünüz gibi; küme parantezlerinin boş bırakılması sonucu boş bir nesne oluşturulabilir. Işte JSON'da en dıştaki küme parantezleri buradan gelmektedir.

Bir JSON nesnesi oluşturulması ya da şöyle söyleyelim; bir nesne oluşturulup o nesnenin işaret ettiği tarafın JSON veri kümesi olması da aşağıdaki gibi sağlanır.

```
var jsonObj = {"id": 42, "city": "Konya"};
```

Gördüğünüz gibi oluşturulan nesnenin işaret ettiği taraf bir JSON veri satırı. Işte JavaScript içerisinde JSON kullanılırken yukarıdaki yol izlenir. Ne demek istediğimi örneklerimizi incelediğinizde daha iyi anlayacaksınız.

Ilk olarak HTML içerisinde gömülü olarak bulunan JavaScript kodları arasında JSON kullanımına bakalım. Örnek olarak basit bir veri kümesi oluşturalım daha sonra da bu JSON veri kümesini bir nesneye bağlayalım ve bir web sayfasında kullanalım. Yine basitten daha karmaşık yapılara doğru gidelim.

Aşağıdaki örneğimizi inceleyelim.

```
<!DOCTYPE html>
<html>
    <head>
```

```
        <title>JSON Example</title>
        <meta charset="UTF-8">
        <script>
            <!-- nesne olusturuldu. -->
            <!-- bu nesnenin isaret ettigi bir json veri satiri -->
            var jsonObject = {
                "name": "Emre CAN",
                "surname": "ÖZTAŞ",
                "job":"Computer Engineer"
                };
        </script>
    </head>
    <body>
        <script>
            document.write(jsonObject.name);
            document.write("<br>");
            document.write(jsonObject.surname);
            document.write("<br>");
            document.write(jsonObject.job);
        </script>
    </body>
</html>
```

Öncelikle yukarıdaki örneğimizi açıklamaya çalışalım. Örneğimizde HTML etiketlerinin arasında gömülü olarak JavaScript kodlarını, `<script></script>` etiketleri arasında yazdık. Bir nesne oluşturduk. Nesnemizin adı: `jsonObject`. jsonObject nesnemizin işaret ettiği taraf JSON veri satırını göstermektedir. Ya da bir JSON veri satırı oluşturduk ve bu veri satırlarını bir nesneye bağladıkta diyebiliriz. Artık bu nesneyi kullanarak JSON veri satırlarını okuyabiliriz veya değiştirebiliriz. Web sayfamızın `<body></body>` kısmında; nesnemizin işaret ettiği JSON verilerini ya da nesnelerini çağırdık ve ekrana yazdırdık. Yazmış olduğumuz örneğimizin ekran çıktısını alalım.

Ekran çıktımızda da görüldüğü üzere, oluşturmuş olduğumuz nesnemizle JSON nesnelerini okuyabiliyoruz. Olaya daha yakından bakalım. Çünkü bu yapı JSON ve JavaScript kullanımında temel bir yapı olacaktır. Eğer bu yapıyı anlarsak; olay bitmiştir.
Öncelikle oluşturmuş olduğumuz JSON veri satırlarımız şöyle idi:

```
{
    "name": "Emre CAN",
    "surname": "ÖZTAŞ",
    "job":"Computer Engineer"
}
```

Daha sonrada jsonObject adından bir nesne oluşturduk ve JSON satırlarımızı bu nesneye bağladık. Herşey bu kadar basit. Şimdilik JSON ile işimiz kalmadı. Sıra bu JSON verilerini okumaya geldi. Okuma işlemini ise JSON satırlarını işaret eden nesnemiz aracılığıyla gerçekleştirdik. Nasıl mı? Aşağıdaki yapımızı inceleyelim.

```
jsonObject.name    // name cagrildi (Emre Can)
jsonObject.surname // surname cagrildi (ÖZTAŞ)
jsonObject.job     // job cagrildi (Computer Engineer)
```

Yukarıdaki yapımızı, yorum satırları ile basitçe açıklamaya çalıştık. Bu tek satırlık veri satırlarında böyledir. Lakin Object veya Array tipinde veri satırlarında alacağımız değerin indis'ini de belirtmemiz gerekiyor ama bu temel şekildir.

Şayet örneğimizi ekrana yazdırmaz yerine bir değişkene de atayabiliriz.

```
var name = jsonObject.name;
var surname =jsonObject.surname;
var job = jsonObject.job;
```

Değişkene atadığımız bu değerleri istediğimiz şekilde kullanabiliriz, artık.

Burada çok küçük bir bilgi vereyim. Değişkene atamasını yaptığımız değerin tipini öğrenmek istersek; JavaScript ortamında `typeof` komutunu kullanmamız yeterli. Aşağıdaki gibi kullanımları mevcuttur.

```
typeof(jsonObject.name)
// veya
typeof jsonObject.name
```

Örneğimiz sadece basit bir veri yapısıydı. Peki örneğin elimizde bir dizi ya object tipinde veri satırları olsa? O zaman bu verileri nasıl çağıracağız? Cevap basit: temel olarak yine yukarıdaki gösterdiğimizde uyguladığımız yöntem gibi.

Elimizde bir veri dizi yapısı olsun. Hatta bir kaç tane dizi yapısı olsun. Ayrıca object tipinde ve normal tipte veri satırları olsun. Bunları oluşturduğumuz nesnemizle çağıralım. Bu örneğimiz buraya kadar anlattıklarımızı kapsayacak nitelikte olacaktır.

Aşağıdaki örneğimizi bakalım.

```
<!DOCTYPE html>
<html>
    <head>
        <title>JSON Example</title>
        <meta charset="UTF-8">
        <script>
            var jsonObject = {
               "tabloAdi": "UyelerVeYoneticiler",
               "hazirlayan": {
                    "hAdSoyad": "Cem IKTA"
                    },
                "uyeler": [
                    {
```

```
                    "uyeAd": "Rozerin",
                    "uyeSoyad": "AKTAŞ"
                    },
                {
                    "uyeAd": "Faruk",
                    "uyeSoyad": "YER"
                    },
                {
                    "uyeAd": "Rıdvan",
                    "uyeSoyad": "KARATAŞ"
                 }
            ],
            "yoneticiler": [
                {
                    "yoneticiAd": "Recep",
                    "yoneticiSoyad": "SAYGILI"
                    },
                {
                    "yoneticiAd": "Anıl",
                    "yoneticiSoyad": "ARAÇ"
                 }
            ]
        };
    </script>
</head>
<body>
    <script>
        document.write("<b>Üyeler</b><br>");
        document.write(jsonObject.uyeler[0].uyeAd + " "
        + jsonObject.uyeler[0].uyeSoyad);
        document.write("<br>");
        document.write(jsonObject.uyeler[1].uyeAd + " "
        + jsonObject.uyeler[1].uyeSoyad);
        document.write("<br>");
        document.write(jsonObject.uyeler[2].uyeAd + " "
        + jsonObject.uyeler[2].uyeSoyad);
        document.write("<br>");
        document.write("<b><br>Yöneticiler</b><br>");
        document.write(jsonObject.yoneticiler[0].yoneticiAd+ " "
        + jsonObject.yoneticiler[0].yoneticiSoyad);
        document.write("<br>");
        document.write(jsonObject.yoneticiler[1].yoneticiAd+ " "
        + jsonObject.yoneticiler[1].yoneticiSoyad);
        document.write("<br>");
        document.write("<br>");
        document.write("<b>Tablo Adı: </b>" +
        jsonObject.tabloAdi);
        document.write("<br>");
        document.write("<b>Hazırlayan: </b>" +
        jsonObject.hazirlayan.hAdSoyad);
    </script>
```

```
    </body>
</html>
```

Yukarıdaki örneğimizi inceleyelim. Yine bir nesnemiz var ve bu nesnemiz JSON veri satırlarını işaret ediyor. JSON veri satırlarımızda iki tane dizimiz var ve bu dizilerimizin de elemanları var. Ayrıca bir tane object tipinde bir tane de normal veri tipinde veri satırımız bulunuyor.
Örneğimizi açıklamadan önce ekran çıktısına bir bakalım.

Yukarıdaki ekran çıktısında da görüldüğü üzere; istenilen dizi, object ve normal veri satırlarını, oluşturulan nesne ile kolayca çağırabiliyoruz.

Normal bir veri satırının okunmasını ilk örneğimizde anlatmıştık. O yüzden bunu pas geçiyorum. Ilk olarak object tipindeki veri satırımıza bakalım.

```
"hazirlayan": {
     "hAdSoyad": "Cem IKTA"
}
```

JSON veri satırları arasındaki object'imiz bu şekildeydi. Peki biz bu object tipindeki değeri nasıl okuduk ya da okuyabiliyoruz. Aşağıdaki yapımıza bakalım.

```
jsonObject.hazirlayan.hAdSoyad
```

Normal bir JSON veri satırını; `nesne.name` şeklinde okuyabiliyorduk. Object tipinde ise bir de o object'in sahip olduğu alanların name'lerini yazmamız gerekir. Yani;

```
nesne.name.inTheObjectName
```

şeklinde olacaktır.

Şimdi gelelim dizi yapısındaki JSON veri satırlarımıza. Öncelikle örneğimizdeki herhangi bir dizi yapısını alalım.

```
"uyeler": [
              {
                    "uyeAd": "Rozerin",
                    "uyeSoyad": "AKTAŞ"
                    },
               {
                     "uyeAd": "Faruk",
                     "uyeSoyad": "YER"
                     },
               {
                     "uyeAd": "Rıdvan",
                     "uyeSoyad": "KARATAŞ"
                     }
                ]
```

Örneğimizde kullandığımız dizimiz bu şekilde idi. Peki bu dizi elemanlarını nasıl çağırabiliyoruz veya çağırdık. Çok basit aşağıdaki yöntemde olduğu gibi.

```
jsonObject.uyeler[indis].uyeAd
jsonObject.uyeler[indis].uyeSoyad
```

Çağıracağımız dizi elemanının indis değerini, varsa dizi içerisinde tanımlı olan name alanını yazmamız yeterli olacaktır. Şayet dizimiz içerisindeki elemanların bir name alanları olmasaydı, o zaman name alanını yazmamız gerekmeyecekti. Yani yapımız:

```
jsonObject.uyeler[indis]
```

Şeklinde olacaktı. Bunu da unutmayalım.
Buraya kadar olan bölümde; JSON ile JavaScript kullanımını anlatmaya çalıştık. Örneklerimizi hep `integer` ve `string` tipindeki veri tipleri üzerine kurduk. Bu demek değildir ki diğer veri tiplerini kullanamayız. Diğer veri tiplerini de dördüncü bölümde bahsetiğimiz şekilde pek ala kullanabiliriz.

Herneyse, kaldığımız yerden devam edelim.

Peki bu JSON verilerini değiştirebilir miyiz? Evet değiştirebiliriz. Aşağıdaki örneğimize bakalım.

```
jsonObject.tabloAdi = “Yeni Tablo”;
jsonObject.hazirlayan.hAdsoyad = “Özcan ACAR”;
jsonObject.uyeler[0].uyeAd = "Cem";
jsonObject.uyeler[0].uyeSoyad = "YILMAZ";
```

Yukarıdaki örneğimizde; JSON veri satırlarında olan alanlarının değerlerini değiştirdik. Bu şekilde diğer elemanların özellikleri de değişebilir. Lakin bu değişim `Runtime (Çalışma Zamanı)`'da yapılan bir değişikliktir. Yani JSON veri satırlarında herhangi bir değişikliğe sebep olmaz. Eğer JSON satırlarını kalıcı olarak değiştirmek istiyorsak; `.json` uzantılı bir dosya oluşturmalı ve bir programlama dili ile değiştirmeliyiz.

Son olarak; örneğimizde, JSON satırlarını, sırasıyla tek tek yazarak ekrana getirdik. Peki elimizde bir dizi ve çok sayıda da veri olduğunu düşünürsek, bu verilerin tümünü daha kısa bir yöntemle getiremez miyiz? Evet getirebiliriz. Basit bir `for` döngüsü ile veri satırlarına ulaşabiliriz. Aşağıdaki örneğimize bakalım.

```
<!DOCTYPE html>
<html>
    <head>
        <title>JSON Example</title>
        <meta charset="UTF-8">
        <script>
            var jsonObject = {
                "uyeler": [
                    {"uyeAd": "Rozerin",
                     "uyeSoyad": "AKTAŞ"},
                    {"uyeAd": "Faruk",
                     "uyeSoyad": "YER"},
                    {"uyeAd": "Rıdvan",
                     "uyeSoyad": "KARATAŞ"}
                ],
                "yoneticiler": [
                    {"yoneticiAd": "Recep",
                     "yoneticiSoyad": "SAYGILI"},
                    {"yoneticiAd": "Anıl",
                     "yoneticiSoyad": "ARAÇ"}
                ]
            };
        </script>
    </head>
    <body>
        <script>
            document.write("<b>Üyeler</b><br>");
            for(i=0; i<jsonObject.uyeler.length;i++){
                document.write(jsonObject.uyeler[i].uyeAd + " ");
                document.write(jsonObject.uyeler[i].uyeSoyad);
                document.write("<br>");
            };
            document.write("<br>");
            document.write("<b>Yöneticiler</b><br>");
          for(i=0; i<jsonObject.yoneticiler.length;i++){
            document.write(jsonObject.yoneticiler[i].yoneticiAd
            +" ");
            document.write(jsonObject.yoneticiler[i].yoneticiSoyad);
            document.write("<br>");
          };
        </script>
    </body>
</html>
```

Yukarıdaki örneğimizde basit 2 tane for döngüsü açtık. Bu for döngülerinde veri satırlarının uzunluklarını:

```
jsonObject.diziAd.length
```

Şeklinde bulabiliriz.

Son örneğimizin ekran çıktısına bakalım.

Yukarıdaki örneğimizde de görüldüğü gibi for döngüsü ile JSON veri satırlarına kolayca ulaşabiliyoruz. Burada for döngüsü yerine while döngüsünü de tercih edebilirdik. Bu tamamen bize kalmış bir durum. Yöntemi bildikten sonra uygulaması kolaydır.

5.1 JavaScript Dosyalarında JSON Kullanımı

Buraya kadar herşey normal, güzel, iyi falan da “Peki, harici bir JavaScript dosyası içerisinde JSON'ı kullanilir miyiz aga?” diye soranınız yok mu? Elbette vardır. Peki efendim hemen cevap vereyim: evet. Kısa ve öz bir cevap oldu. Bu bölümde çok soru sordum, biliyorum. Hakkınızı helal edin.

Harici JavaScript dosyası içerisinde JSON'ı kullanabiliriz. Çünkü JSON zaten JavaScript tabanlı. Bildiğiniz gibi harici JavaScript dosyalarının uzantıları `.js` idir. Gelin bir örnek yapalım. Harici bir JavaScript dosyası açalım. Bu dosyanın adını, yazacağımız örneğimize uygun olarak: `jsonColorChancer` olarak belirleyelim. Yazacağımız örneğimizde; kullanıcının tıklayacağı butona göre sayfamızın arkaplanı değişsin. 3 temel rengi yani `RGB (Red – Green – Blue)` JSON veri satırlarında belirtelim. Daha sonrada yazacağımız bir fonksiyonla sayfanın arkaplanını değiştirelim.

Ilk olarak `jsoncolorchancer.js` dosyamızı oluşturalım. Oluşturacağımız dosyanın içeriği aşağıdaki gibi olacaktır.

`jsoncolorchancer.js`

```
var cssObject = {
            "color": [
                        "red",
                        "green",
                        "blue"
                        ]
};

function changeColor(i) {
    document.getElementById("myForm").style.backgroundColor =
    cssObject.color[i];
}
```

Yukarıdaki örnek dosyamızda üzerinde biraz duralım. Ilk olarak bir `Object (Nesne)` oluşturduk. Oluşturmuş olduğumuz bu nesnemize de 3 temel rengi, `color` dizisinin elemanları olarak tanımladık. Daha sonra; `changeColor(i)` isimli bir fonksiyon oluşturduk. Bu fonksiyon kendisine gelen değere göre JSON

veri satırlarındaki renkleri seçecek ve formun arkaplanını değiştirecektir.

Şimdi de HTML kodlarımızı yazalım.

`index.html`

```
<html>
    <head>
        <title>JSON Example</title>
        <meta charset="utf-8">
        <script src="jsonColorChanger.js"></script>
    </head>
    <body id="myForm">
        <input type="button" value="Red" onclick="changeColor(0)">
        <input type="button" value="Green" onclick="changeColor(1)">
        <input type="button" value="Blue" onclick="changeColor(2)">
    </body>
</html>
```

Oluşturmuş olduğumuş HTML sayfamızda 3 tane buton var. Bu butonların, `onclick Event(Olay)`'ine, sayfamıza eklediğimiz harici JavaScript dosyasında tanımlı olan fonksiyonu yazdık. Yani kısaca kullanıcı seçtiği butona göre arkaplan değişecek.

Yukarıdaki oluşturmuş olduğumuz sayfamızın ekran çıktısına bakalım ve doğru yazmış mıyız test edelim. Ekran çıktımız aşağıdaki gibi olacaktır.

Yukarıdaki ekran çıktısında da görüldüğü üzere; JSON'ı HTML etiketleri arasında gömülü kullanılan JavaScript içerisinde kullanabileceğimiz gibi harici bir JavaScript dosyasında da kullanabiliriz.

JSON, çok kullanışlı ve esnek bir yapıya sahip. Bu ve buna benzer ya da çok daha farklı bir amaçla kullanabilirsiniz. Size son bir tüyo vereyim. JSON son zamanlarda çeşitli programlarda veya web sayfalarında; ayar dosyalarının tutulmasında oldukça sık kullanılıyor.

5.2 Extras

`Extras (Ekstralar)` kısmında; JSON ile kullanılabilecek olan temel JavaScript metotlarına değineceğiz. Bu metotları kullanmamızın amacı değişmekle birlikte temelde daha iyi bir JSON gösterimi veya kullanımı sağlamak desek yanlış olmaz herhalde.

5.2.1 parse( )

`parse(text, reviver)` fonksiyonu, text olarak yazılmış olan JSON veri kümesini JavaScript nesnesinin kullanabileceği hale getiren bir fonksiyondur. Ya da daha genel bir tabirle; JSON formatına çeviren bir fonksiyondur. Burada `text` parametresi; text olarak yazılmış olan bir JSON ifadesidir. `reviver` parametresi ise text ifadesi üzerinde belli işlemleri gerçekleştirmek için tanımlanabilecek fonksiyonları içeren parametredir. Dilerseniz kullanmak zorunda değilsiniz. Biz de örneklerimizde reviver parametresini kullanmayacağız.

parse() fonksiyonunun kullanımı aşağıdaki gibidir.

```
JSON.parse(text, reviver)
```

Burada JSON, JavaScript'teki bir `class (sınıf)` yapısı olarak algılanabilir, doğrudur. JSON, JavaScript'ten türemiş bir yapıdır. Yani JSON'a JavaScript'in bir alt kümesi desek yanlış olmaz herhalde. Sözün özü; JavaScript ortamında JSON ile işlem yapılacaksa, JSON sınıfı kullanılmalıdır.

Basit örneklerle başlayalım. Örneğin aşağıdaki gibi bir JSON yapımız olsun.

```
'{"id": 42, "city": "Konya"}'
```

Yukarıdaki yapımızı bir nesneye bağlayalım.

```
var jsonObj = '{"id": 42, "city": "Konya"}';
```

Görüldüğü gibi text yapısında bir JSON formatı. “Aga neden böyle bir şey yapayım ki? JSON veri formatında yazarım her türlü!” demeyin dostlar. Örneğin bazı durumlarda kendiniz JSON veri satırları oluşturmak isteyebilirsiniz. Farz-ı misal; kullanıcının gireceği değerlere göre bir JSON yapısı oluşturmak isteyebilirsiniz. Böyle durumlarda gelen string değerler ile JSON veri satırlarını oluşturmak sıkıntılıdır benden söylemesi. Işte bu gibi durumlarda parse( ) fonksiyonunu kullanabilirsiniz.

Son yazdığımız satırımızı parse edelim.

```
var convertToJson = JSON.parse(jsonObj);
```

`convertToJson` nesnesi, `jsonObj` nesnesini parse etti. Yani text yapısını, JSON'a çevirdi. Artık convertToJson nesnesini dilediğimiz gibi kullanabiliriz.
Örneğimizi HTML sayfamıza yerleştirelim ve kullanalım.

```
<html>
    <head>
        <title>JSON Example</title>
        <meta charset="utf-8">
        <script>
            var jsonObj = '{"id": 42, "city": "Konya"}';
            var convertToJson = JSON.parse(jsonObj);
        </script>
    </head>
    <body>
```

```
        <script>
            document.write("id: " + convertToJson.id);
            document.write("<br>");
            document.write("city: " + convertToJson.city);
        </script>
    </body>
</html>
```

Yukarıdaki örneğimizi çalıştırdığımızda; `id` ve `city` değerleri, JSON veri satırlarından çekilip ekrana yazdırılacaktır. Eğer JSON satırlarını parse etmez isek ekranda herhangi bir ifade çıkmaz. Ya da şöyle söyleyim; JavaScript parse edilmemiş satırların JSON veri satırı olduğunu anlayamaz.

Bu yapıyı yani parse( ) fonksiyonunu kullanarak; `' '` içerisinde verilen her türlü JSON satırları doğrudan JSON formatına çevrilir.

5.2.2 eval( )

`eval( )` fonksiyonu, `JSON.parse(text)` fonksiyonunu deskteklemeyen tarayıcılar için kullanılan bir fonksiyondur. JSON yeni bir teknoloji ve hala günümüzde eski tarayıcılar kullananlar var. İşte böyle durumlarda parse etmek yerine eval( ) fonksiyonunu kullanabilirsiniz.

eval( ) fonksiyonunun kullanımı aşağıdaki gibidir.

```
eval("(" + objectName +")")
```

Kullanımda ki `objectName` parametesi, JSON veri satırlarını tutan nesnenin adıdır. Diğer önemli bir noktada parantez ve tırnakların kullanımı zorunludur. Yani kısacası kullanımı yukarıdaki ifadeyle aynı olmalıdır.

parse( ) alt başlığında yaptığımız örneğimizi eval ( ) fonksiyonu ile yapmak istersek, aşağıdaki gibi bir yapı elde etmiş oluruz.

```
var jsonObj = '{"id": 42, "city": "Konya"}';
var convertToJson = eval("(" + jsonObj + ")");
```

Text'ten JSON'a dönüştürme işlemi tamamlandı artık; convertToJson nesnesini dilediğimiz gibi kullanabiliriz. parse( ) alt başlığındaki örneğimizi yeniden yazmak istersek aşağıdaki gibi olacaktır.

```
<html>
    <head>
        <title>JSON Example</title>
        <meta charset="utf-8">
        <script>
            var jsonObj = '{"id": 42, "city": "Konya"}';
            var convertToJson = eval("(" + jsonObj + ")");
        </script>
    </head>
    <body>
        <script>
```

```
            document.write("id: " + convertToJson.id);
            document.write("<br>");
            document.write("city: " + convertToJson.city);
        </script>
    </body>
</html>
```

5.2.3 stringify( )

`parse()` ve `eval()` fonksiyonları, kendisine parametre olarak gönderilen text ifadeleri JSON formatına çeviriyordu, değil mi? Işte `stringify()` fonksiyonu da kendisine parametre gönderilen JavaScript değerlerini JSON formatına çeviren bir yapı. Şimdi burada aklınıza şu soru gelebilir. Gelmez demeyin stackoverflow'da bile sormuşlar, baktım. Herneyse işte şu soru: “Aga şimdi bu parse, eval'i falan anladıkta stringify fonksiyonu da aynı işi yapmıyor mu?”. Hayır agası yapmıyor. parse( ) ve eval( ) fonksiyonları; text olarak yazılmış JSON veri satırlarını, bir kaç küçük işareti kaldırarak JSON formatına çeviriyordu ki yazdığımız değerler JSON formatına sahipti, eğer dikkatli takip ettiyseniz. stringify( ) fonksiyonu ise kendisine parametre olarak gönderilen JavaScript değerlerini doğrudan JSON formatına çeviriyor.

stringify( ) fonksiyonunun kullanımı aşağıdaki gibidir.

```
JSON.stringify(objectName);
```

Burada objectName parametresini, JavaScript ile belirleyeceğiz. Yani bizim belirlediğimiz belli değerleri JSON formatına çevireceğiz. Öncelikle bir Object tipinde nesne oluşturmamız gerekli. Daha sonra vereceğimiz bu parametreyi, stringify( ) fonksiyonu ile JSON formatına çevireceğiz.

Adım adım gidelim. Ilk olarak Object tipinde bir nesne oluşturalım.

```
var jsonObj = new Object();
```

Bu nesnemize belli değerler belirleyelim.

```
jsonObj.textType = "James Sawyer";
jsonObj.arrayType = ["one", "two", "three"];
jsonObj.booleanType = false;
jsonObj.nullType = null;
jsonObj.numberType = 3.14;
```

Şimdi de bu nesnemizi JSON formatına çevirelim.

```
var jsonFormat = JSON.stringify(jsonObj);
```

Görüldüğü gibi jsonFormat nesnesi artık JSON formatında bir yapıya dönüşmüş oldu. Buraya kadar olan satırlarımızı bir örnek haline getirirsek, örneğimiz aşağıdaki gibi olacaktır.

```
<html>
    <head>
        <title>JSON Example</title>
        <meta charset="utf-8">
```

```
    </head>
    <body>
        <script>
            var jsonObj = new Object();
            jsonObj.textType = "James Sawyer";
            jsonObj.arrayType = ["one", "two", "three"];
            jsonObj.booleanType = false;
            jsonObj.nullType = null;
            jsonObj.numberType = 3.14;
            var jsonFormat = JSON.stringify(jsonObj);
            document.write(jsonFormat);
        </script>
    </body>
</html>
```

Bu örneği tarayıcıda görüntünlediğimiz zaman aşağıdaki çıktıyı verecektir.

```
{"textType":"James Sawyer","arrayType":
["one","two","three"],"booleanType":false,"nullType":null,"numberType":3.
14}
```

Görüldüğü stringify( ) fonksiyonu çok kullanışlı bir yapıya sahip. Istenilen her değeri doğrudan JSON formatına çevirebilmektedir. Burada değinmek istediğim son bir konu var. Object tipinde bir nesne oluşturarak yukarıdaki gibi bir çıktı elde edebiliyoruz. Bunun dışında bir dizi değişken oluşturarak, sadece dizi elemanlarından oluşan bir JSON formatı da elde edebiliriz. Aşağıdaki örneğimizi inceleyelim.

```
var arrayObj = new Array();
arrayObj[0] = "zero";
arrayObj[1] = "one";
arrayObj[2] = "two";
arrayObj[3] = "three";
arrayObj[4] = "four";
var jsonFormat = JSON.stringify(arrayObj);
document.write(jsonFormat);
```

Yukarıdaki örneğimizin çıktısı da aşağıdaki gibi olacaktır.

```
["zero","one","two","three","four"]
```

BÖLÜM 6:
Schema

JSON'ın en önemli kullanım alanına geldik. Bu bölümde oldukça işinize yarayacak, hatta JSON'ın temel kullanım alanı diyebileceğimiz bir özelliğini öğreneceğiz. Bölüm başlığında da belirtildiği gibi `Schema(Şema)` yapısı üzerinde duracağız.

Öncelikle Schema nedir? Bu sorunun cevabını aramakla işe başlayalım. Bildiğiniz gibi XML için de veri taşıma formatıdır dedik ve JSON'ı XML'in alternatifi olarak gösterdik. Aynı yapı XML'de de bulunmaktadır. Şayet XML biliyorsanız, birazdan anlatacaklarımız size yabancı gelmeyecektir.

JSON Schema, var olan `data (veri)` yapınızı tanımlamanızı sağlayan bir yoldur. Bu da nedir diyebilirsiniz. Hemen açıklayalım. JSON dosyalarının uyması gereken veya daha genel bir tabirle olması gereken şekilde kullanılması için oluşturulan bir yapıdır. Oluşturulan JSON dosyasının geçerli olup olmadığı schema kullanılarak kolayca öğrenilebilir. Bu yapı dilden bağımsız olacağı için: tanımlanan programlama dilinde hangi nesnelerin olacağı, nesnelerin ne tür üyeleri olacağı, bir nesnenin içinde hangi nesneler olabileceği gibi kurallar schema ile tanımlanır.

Daha iyi anlamanız için basit bir örnek yapalım. Örneğin elimizde aşağıdaki gibi bir JSON dosyası olsun.

```
{
    "id": 12,
    "name": "Car",
    "price": 20.000,
    "color": ["red", "green", "yellow"]
}
```

Yukarıdaki örneğimiz basit bir örnek gibi duruyor. Lakin yukarıdaki satırları bir programcının anlaması daha doğrusu anlayıp uygulaması zordur. Neden? Çünkü id ne olacak? Price, hangi aralıklarda belirlenebilir? Örneğin 0 yani bedava olabilir mi? Sadece 3 renk mi var? Bu gibi soruların yanıtını bilmesi gerekiyor. Ama bu soruların cevabını sadece bu satırları yazan birisi bilebilir. O yüzden JSON Schema kullanmak bu ve buna benzer durumlarda çok gereklidir.

Schema'nın anlamını veya görevini bulduğumuza göre konumuza kaldığımız yerden devam edelim. Çeşitli programlama dillerinde kullanılabilecek olan bu yapı için çeşitli `Schema Validation Libraries (Şema Doğrulama Kütüphaneleri)` bulunmaktadır. Aşağıdaki adreste, `Validator` başlığı altında çeşitli programlama dilleri için kullanılabilecek olan doğrulama kütüphaneleri verilmiştir.

```
http://json-schema.org/implementations.html
```

Bu doğrulama kütüphaneleri kullanılarak basit şekilde, yazılan satırların schema doğrulaması gerçekleştirilebilir.

Bu bölümde temel olarak bir JSON Schema nasıl oluşturulur, bunun üzerinde duracağız.

Öncelikle bir schema yapısı nasıl oluşturulur buna bakalım. Her JSON Schema dosyası aşağıdaki satırla başlamak zorundadır.

```
"$schema": "http://json-schema.org/draft-04/schema#"
```

Burada belirtilen `$schema`, bu dosyanın bir schema olduğunu ve `draft-04`'e göre oluşturulduğunu tanımlar. $shema'dan sonra bir adres verilmiştir. Işte bu adresin içerisindeki draft-04 bu schema'nın versiyonunu belirlemektedir. Ayrıca yukarıdaki adrese gittiğinizde bir dosya indirilecektir. Işte o dosyanın içeriği JSON Schema'da tanımlanan schema yapısını içermektedir. Dosyanın içeriğini biraz incelemenizi tavsiye ederim. Ekstra olarak; draft-00, draft-01, draft-02 ve draft-03'e de bakabilirsiniz. Bu adreslerin tümüne gittiğinizde o draft'a göre düzenlenmiş bir dosya indirilecektir. Yani kısaca bir versiyon numarasıdır.

Devam edelim. Indirilen dosyada, aşağıdaki yapılardan bazıları mevcuttur, incelerseniz. Schema oluştururken bu yapıları ya da üyeleri kullanmalısınız.

`title`	Başlık tanımlamak için kullanılır.
`description`	Açıklama yazmak için kullanılır.
`type`	JSON nesnesi tanımlanması için kullanılır.
`properties`	JSON dosyasında kullanılmak üzere; maksimum-minimum aralıkları, değer türleri v.s tanımlamak için kullanılır.
`required`	Kullanılması gerekli alanların listesini tanımlamak için kullanılır.
`minimum`	Minimum değer tanımlamak için kullanılır.
`exclusiveMinimum`	Bir değerin minimum değeri olup olmayacağını belirtir. Boolean bir değer almak zorundadır.
`Maximum`	Maksimum değer tanımlamak için kullanılır.
`ExclusiveMaxsimum`	Bir değerin maksimum değeri olup olmayacağını belirtir. Boolean bir değer almak zorundadır.
`maxLength`	Maksimum büyüklüğü belirtmek için kullanılır.
`minLength`	Minimum büyüklüğü belirtmek için kullanılır.

Kaldığımız yerden devam edelim.

```
{
    "title": "title",
    "description": "A good description",
    "type": "object"
}
```

Yukarıdaki alanlarda bir JSON schema'da olması gereken alanlardır. Zorunlu değildir. Fakat yazılması dosyanın okunabilirliği artmaktadır.

Buraya kadar eğer bir sıkıntı yoksa basit bir JSON Schema oluşturalım ve bu schema'ya göre nasıl değerler yazmamız gerektiği hakkında basit bir örnek yapalım.

Ilk olarak JSON schema'dan başlayalım. Örnek JSON schema'mız aşağıdaki gibi olsun.

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "This is JSON Schema about car company",
    "description": "Car models",
    "type": "object",
    "properties": {
        "id": {
            "description": "The unique identifier for every car",
            "type": "integer"
        },
        "name": {
            "description": "Name of car",
            "type": "string"
        },
        "model": {
            "description": "Model of car",
            "type": "string"
        },
        "price": {
            "type": "number",
            "minimum": 0,
            "exclusiveMinimum": true
        },
        "color": {
            "description": "Color of car",
            "type": "array"
        },
        "tags": {
            "description": "Car tags",
            "type": "array"
        },
        "required": ["id", "name", "price"]
    }
}
```

Yukarıdaki JSON Schema'mızı inceleyelim. Öncelikle bu dosyanın bir JSON Schema'sı olduğunu belirttik. Daha sonra; `title`, `description` ve `type` nesnelerinde gerekli açıklamaları yaptık. Diğer alanlar ise bizim kendi datamızın yapısını oluşturmakta. Bu örneğimizde bir araba satış firmasının data yapısını oluşturmaya çalıştık. Dikkat ederseniz datamızda bulunan tüm alanları yazdık ve bir description yani tanım belirttik. Buraya kadar herhangi bir sıkıntı yok. Bu belirttiğimiz datalarımızın tiplerine dikkati çekmek isterim. Gördüğünüz gibi her datanın daha doğrusu her `attribute (özellik)`'yi tırnak işaretleri (“ “) arasında yazdık. Bu genel itibariyle böyledir. Yani attribute'nin tipi yazılacağı zaman tırnak işaretleri arasında belirtilmesi zorunludur. Buna dikkat edelim. Son satırmızda ise; `required (gerekli)` alanını açtık ve

hangi alanların required olması gerektiğini söyledik. Buradaki required anahtar kelimesi, bahsettiğimiz; draft-04 dökümanında geçen özel bir kelime. Bu kelime ile hangi alanların gerekli olduğunu ya da bir anlamda hangi alanların boş geçilemeyeceğini belirttik. Yani bu alanlar kesinlikle boş geçilemeyecek.

Devam edelim. Elimizde JSON Schema'sı var. şimdi bu schema'ya göre datamızı oluşturalım. Örneğimiz aşağıdaki gibi olacaktır.

```
[
    {
        "id": 1,
        "name": "Audi",
        "model": "A4 2.0 TDI 150 hp multitronic",
        "price": 156.935,
        "color": ["black", "metalic gray", "wite"],
        "tags": ["faster", "comfortable", "safety"]
    },
    {
        "id": 2,
        "name": "Audi",
        "model": "A7 Sportback 2.0 TFSI 252 hp quattro S tronic",
        "price": 324.224,
        "color": ["metalic gray", "wite"],
        "tags": ["faster", "comfortable", "safety", "swanky"]
    }
]
```

JSON Schema'ya göre oluşturmuş olduğumuz örneğimiz yukarıdaki gibidir. Gerekli alanları doldurduk. Yani bize verilen JSON Schema ya da buna bir sözleşme dersek; bu sözleşmedeki ilgili yükümlülükleri yerine getirdik. Burada aklınıza takılacak olan soru: “Aga neden array (dizi) olarak veri satırlarını tanımladık?” hemen açıklayım agası. Bize verilen JSON Schema yapısına bakarsanız; properties başlığı bir dizi olarak tanımlanmış. Yani properties adında bir dizimiz var ve bu diziminde elemanları var. yani bir nevi veri tabanındaki veri satırları olarak düşünebilirsiniz. Bizden istediği sadece bu veri satırlarına, veri tipine uygun datalar yerleştirmemiz. Işte bu yüzden dizi şeklinde alanlarımızı doldurduk. Bilmem anlatabildim mi?

JSON Schema çok kullanışlı ve şık bir yapıya sahip. Bu yapıyı kullanarak çeşitli işlemler gerçekleştirebilirsiniz. Zaten JSON'ın genel kullanım amacı da bu yöndedir. Öğrenmenizde hatta çok iyi öğrenmenizde çok büyük yarar olacaktır. Benden söylemesi.

JSON Schema hakkında daha detaylı bilgi edinmek isterseniz aşağıdaki adrese bir göz atmanızı tavsiye ederim.

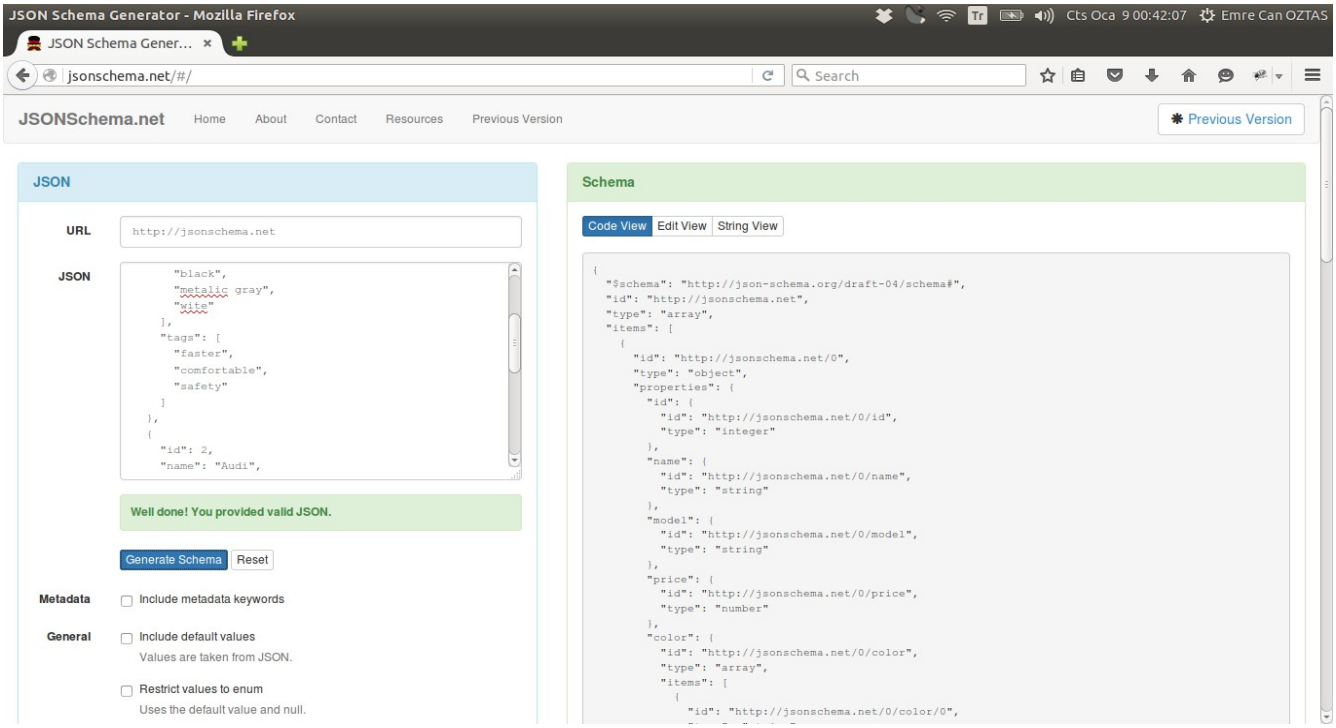
```
http://json-schema.org/
```

Bunun dışında bir çok editör, JSON Schema yapısını tanımamakla birlikte; hata gösterimi veya satırların renklendirilmesi gibi işlemleri yerine getiremiyor. Bunun yerine online olarak kullanacağınız bir editörden bahsetmek isterim.

Öncelikle aşağıdaki adresi ziyaret edelim.

```
http://jsonschema.net/#/
```

Yukarıdaki adrese gittiğimizde, bizi aşağıdaki sayfa karşılayacaktır.



Açılan bu sayfada; normal JSON veri satırlarınızı yazarak bu veri satırlarının JSON Schema karşılığını alabilirsiniz. Oldukça kullanışlı. Kesinlikle kullanmanızı tavsiye ederim. Sol taraftaki JSON alanına; JSON veri satırlarınızı yazıp hemen altında bulunan `Generate Schema` butonuna tıkladığınızda, sağ taraftaki alanda JSON Schema yapısı oluşturulacaktır. Oluşturmuş olduğumuz örneğimizi denediğimizde sağ taraftaki karşılığına uyduğunu görebiliyoruz. Burada da required v.s gibi alanları seçerekte schema'ya ekleyebilirsiniz. Daha sonra bu schema'yı alıp kullanabilirsiniz.

BÖLÜM 7:
PHP & JSON

Bu bölümde temel olarak `PHP` ile `JSON` kullanımından bahsedeceğiz. Neden PHP? Neden başka bir programlama dili değil? Çünkü PHP'de JSON işlemleri basit ve PHP, JSON'a tam destek verdiği için PHP'yi seçtim. Aslında PHP genel anlamda basit bir dil. Eğer web programlama yapmak istiyorsanız, ya da web için bir programlama dili arıyorsanız kesinlikle PHP. Benim gibi freelancer'lar içinde PHP'nin yerini hiç birşey tutmuyor. Bilmiyorsanız; kesinlikle öğrenin derim ki zaten öğrenmesi en fazla 2 haftanızı alır (Ben iki haftada öğrenmiştim). PHP biliyorsanız ve yeni bir dil arıyorsanız; Ruby'e bir bakın derim.

Lafı fazla uzatmadan size PHP hakkında son bir bilgi vereyim. Artık GTK kullanılarak, PHP ile masaüstü programcılık bile yapılabiliyor. Düşünün devir ne kadar ilerledi. Eğer ilginizi çektiyse aşağıdaki sayfayı incelemenizi öneririm.

http://gtk.php.net/

Herneyse konumuza geri dönelim.

7.1 PHP ile JSON Kullanımı

PHP ile JSON işlemleri son derece basittir. Bölümün başında da belirttiğim gibi ekstra bir kurulum, plugin veya API'ye gerek yoktur. `PHP 5.2` sürümünden itibaren JSON'ı desteklemeye başlamıştır. Hazırsanız PHP'nin sahip olduğu fonksiyonları kullanarak JSON işlemlerine başlayalım.

PHP ortamında çalışmak için: `htdocs (XAMPP, LAMPP veya MAMPP)` veya `www (EasyPHP veya Wamp Server)` yani `localhost`'a bir tane çalışma dosyası oluşturalım ve `.php` uzantılı bir dosya oluşturalım. Örneklerimizi bu PHP dosyası üzerinde anlatmaya çalışacağız.

7.1.1 json_encode( )

`json_encode()` fonksiyonu, PHP ile yazılmış veri satırlarını JSON formatına çevirir. json_encode( ) kullanımı aşağıdaki gibidir.

```
json_encode(parametre)
// degiskene baglasak iyi olacaktır.
$jsonResult = json_encode(parametre);
```

PHP ortamında pek çok kullanım şekli olmakla birlikte biz en çok kullanılan stiller üzerinde durmaya çalışacağız. Örneklerimizi basit tutmak yerine genel kullanım üzerine kuracağız. Ilk olarak JSON veri satırlarını oluşturabilmemiz için PHP'deki `array (dizi)` yapısını bilmemiz gerekecektir. Isterseniz öncelikle PHP'de bir dizi nasıl oluşturulur buna değinelim.
PHP'de bir dizi oluşturmak için aşağıdaki yöntem uygulanır.

```
$arrayName = array();
```

Dilerseniz bu diziye ilk değerlerini atayabilirsiniz.

```
$arrayName = array("emre can", "oztas");
```

PHP'de bir de `Associative Array (İlişkilendirilebilir Dizi)` yapısı vardır. Işte bu yapıda aşağıdaki gibidir.

```
$arrayName = array("name" => "emre", "surname"=> "oztas");
```

Yukarıdaki Associative Array yapısı size bir yerden tanıdık geliyor mu? JSON veri satırlarımıza çok benziyor değil mi? Bu yapı PHP'de yıllardır olan bir yapı yani JSON ile bir alakası yok. Ama JSON yapısı Associative Array'e çok benziyor. Tabi bu da bizim işimize geliyor.

Devam edelim. Örneğin Associative Array ile bir yapı kuralım. Daha sonra json_encode( ) fonksiyonu ile bu yapıyı JSON formatına dönüştürelim. Aşağıdaki örneğimizi inceleyelim.

```
<?php
    # associative array olusturalim.
    $personalInfo = array("name" => "emre can",
                          "surname" => "oztas",
                          "city" => "ankara",
                          "number" => 639,
                          "status" => true,
                          "addres" => null
                          );
    # array'imizi JSON'a cevirelim
    $jsonEncode = json_encode($personalInfo);
    # olusan yapiya bakalim.
    print_r($jsonEncode);
?>
```

Örneğimizi yazdık. Örneğimizdeki satırların üzerine açıklamalarını da yazdık. Şimdi localhost'a yazdıklarımızı görüntüleyelim. Ekran çıktımız aşağıdaki gibi olacaktır.

```
{"name":"emre
can","surname":"oztas","city":"ankara","number":639,"status":true,"addres":null
}
```

Yukarıdaki ekran çıktısında da görüldüğü gibi PHP ile yazmış olduğumuz dizimiz doğrudan JSON formatına çevrildi. Çok basit bir işlem değil mi? Burada unuttuğumuz bir veri tipi var: `array (dizi)`. JSON'da bulunan tüm veri tiplerini gördük lakin diziye değinmedik. O zaman bir de PHP satırlarımıza daha doğrusu oluşturmuş olduğumuz dizimize bir dizi daha ekleyelim. Örneğimizin son şekli aşağıdaki gibi olacaktır.

```
<?php

$personalInfo = array("name" => "emre can",
    "surname" => "oztas",
```

```
    "city" => "ankara",
    "number" => 639,
    "status" => true,
    "addres" => null,
    "skills" => array("JavaSE", "JAVAEE", "GNU-Linux", "PHP",
                      "AngularJS", "Python")

        # yukaridaki dizimiz belirli alanlara sahipse asagidaki
        # sekilde de yazabiliriz.
        /*
         * "skills" => array("skill1" => "JavaSE",
         *                   "skill2" => "JAVAEE",
         *                   "skill3" => "GNU-Linux",
         *                   "skill3" => "PHP",
         *                   "skill4" => "AngularJS",
         *                   "skill5" => "Python")
         */
);
$jsonEncode = json_encode($personalInfo);
print_r($jsonEncode);
?>
```

Dizimizi de örneğimize ekledikten sonra ekran çıktısı aşağıdaki gibi olacaktır.

```
{"name":"emre
can","surname":"oztas","city":"ankara","number":639,"status":true,"addres":null
,"skills":["JavaSE","JAVAEE","GNU-Linux","PHP","AngularJS","Python"]}
```

Görüldüğü gibi dizimiz de JSON formatına dönüştürüldü.

JSON basit bir yapıya sahip. PHP ile kullanımı da oldukça basit, sizde gördünüz. Bu başlığın başında belirtmiştik. PHP'de farklı kullanım stilleri var. İsterseniz bir de `class (sınıf)` yapısını kullanarak oluşturulan JSON yapısına bakalım. Öncelikle PHP'de bir sınıf nasıl oluşturulur önce buna bakalım.

PHP'de bir sınıf aşağıdaki şekilde oluşturulur.

```
class ClassName{

}
```

Yazılan sınıftan bir nesne de aşağıdaki şekilde oluşturulur.

```
$objectName = new ClassName()
```

Bu sınıfın üye değişkenleri de olabilir. Eğer Java veya C# bilginiz varsa; public, private, protected yapılarını biliyorsunuz demektir. Aynı şey PHP içinde geçerlidir. Lakin biz basit bir örnek üzerinde duracağımız için şimdilik sadece public anahtat kelimesini kullanacağız. Aslında var anahtar kelimesini de kullanabiliriz fakat var anahtar kelimesi yakın bir zamanda deprecated (önerilmemek ya da kaldırılmak anlamında) oldu. O yüzden örneğimiz için public kullanmak daha evla duruyor. Herneyse örneğimize geçelim.

Yukarıdaki yazmış olduğumuz örneğimizi sınıf yapısını kullanarak yeniden yazalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<?php

class VarClass {
    # uye degiskenlerimizi tanımlayalim.
    public $name;
    public $surname;
    public $city;
    public $number;
    public $status;
    public $adress = array();
}

    # VarClass'indan bir nesne olusturalim.
    $personalInfo = new VarClass();
    # nesnemizin ozelliklerini girelim.
    $personalInfo->name = "emre can";
    $personalInfo->surname = "oztas";
    $personalInfo->city = "ankara";
    $personalInfo->number = 639;
    $personalInfo->status = true;
    $personalInfo->adress[] = "JavaSE";
    $personalInfo->adress[] = "JavaEE";
    $personalInfo->adress[] = "GNU-Linux";
    $personalInfo->adress[] = "PHP";
    $personalInfo->adress[] = "AngularJS";
    $personalInfo->adress[] = "Python";

    # nesnemiz artik 1 degerler kumesini sakliyor.
    # JSON formatina cevirelim.
    $jsonVar = json_encode($personalInfo);
    print_r($jsonVar);

?>
```

Örneğimizi yazdık. Gördüğünüz gibi bir sınıf tanımladık ve bu sınıfa üye değişkenler atadık. Daha sonra tanımlamış olduğumuz sınıftan bir nesne oluşturduk ve sırasıyla değişkenlerin değerlerini atadık. Daha sonra nesnemizi JSON formatına çevirdik. Bu örneğimizin ekran çıktısı aşağıdaki gibi olacaktır.

```
{"name":"emre
can","surname":"oztas","city":"ankara","number":639,"status":true,"adress":
["JavaSE","JavaEE","GNU-Linux","PHP","AngularJS","Python"]}
```

Ekran çıktımızda da gördüğünüz gibi ilk yaptığımız örnekle aynı çıktıyı elde ettik. Bu iki yapı arasından seçim yapmak size kalmış. PHP > JSON formatında bir sıkıntımız kalmadığına göre bir de tam tersi duruma yani JSON formatından PHP formatına çevirme işlemine bakalım. O işlemde oldukça kolay, göreceksiniz.

7.1.2 json_decode( )

`json_decode()` fonksiyonu, JSON formatındaki verileri PHP yapısına çevirir. Yani bir anlamda geri dönüşüm yapar diyebiliriz.

json_decode( ) fonksiyonunun kullanımı aşağıdaki gibidir.

```
json_decode(parametre)
// degiskene baglasak iyi olacaktır.
$jsonResult = json_decode(parametre);
```

json_encode( ) başlığında yapmış olduğumuz örneğimizi json_decode( ) fonksiyonuna uyarlayalım. Hatta gelin isterseniz, daha önce kullancığımız örneklerimize json_decode( ) fonksiyonunu ekleyelim. Ilk örneğimizle başlayalım.

```
<?php
    $personalInfo = array("name" => "emre can",
                          "surname" => "oztas",
                          "city" => "ankara",
                          "number" => 639,
                          "status" => true,
                          "addres" => null,
                          "skills" => array("JavaSE", "JAVAEE", "GNU-
Linux", "PHP", "AngularJS", "Python")
        );
    # PHP > JSON
    $jsonEncode = json_encode($personalInfo);
    echo "PHP > JSON\n";
    print_r($jsonEncode);
    echo "\n";

    # JSON > PHP
    $jsonDecode = json_decode($jsonEncode);
    echo "JSON > PHP\n";
    print_r($jsonDecode);
?>
```

Yukarıdaki örneğimizin ekran çıktısını alalım.

```
PHP > JSON {"name":"emre
can","surname":"oztas","city":"ankara","number":639,"status":true,"addres":null
,"skills":["JavaSE","JAVAEE","GNU-Linux","PHP","AngularJS","Python"]} JSON >
PHP stdClass Object ( [name] => emre can [surname] => oztas [city] => ankara
[number] => 639 [status] => 1 [addres] => [skills] => Array ( [0] => JavaSE [1]
=> JAVAEE [2] => GNU-Linux [3] => PHP [4] => AngularJS [5] => Python ) )
```

Ekran çıktısında da görüldüğü gibi daha önce yazmış olduğumuz PHP > JSON dönüşüm işlemini JSON > PHP şeklinde tekrar yazdık.

Yapmış olduğumuz ikinci örneğimize de yine aynı şekilde json_decode( ) fonksiyonunu ekleyelim. Örneğimizin son şekli aşağıdaki gibi olacaktır.

```
<?php

class VarClass {
    # uye degiskenlerimizi tanımlayalim.
    public $name;
    public $surname;
    public $city;
    public $number;
    public $status;
    public $adress = array();
}

    $personalInfo = new VarClass();

    $personalInfo->name = "emre can";
    $personalInfo->surname = "oztas";
    $personalInfo->city = "ankara";
    $personalInfo->number = 639;
    $personalInfo->status = true;
    $personalInfo->adress[] = "JavaSE";
    $personalInfo->adress[] = "JavaEE";
    $personalInfo->adress[] = "GNU-Linux";
    $personalInfo->adress[] = "PHP";
    $personalInfo->adress[] = "AngularJS";
    $personalInfo->adress[] = "Python";

    # PHP > JSON
    $jsonVar = json_encode($personalInfo);
    print_r($jsonVar);

    #JSON > PHP
    $phpVar = json_decode($jsonVar);
    print_r($phpVar);

?>
```

Yukarıdaki örneğimize ait olan ekran çıktısı da aşağıdaki gibi olacaktır.

```
{"name":"emre
can","surname":"oztas","city":"ankara","number":639,"status":true,"adress":
["JavaSE","JavaEE","GNU-Linux","PHP","AngularJS","Python"]}stdClass Object
( [name] => emre can [surname] => oztas [city] => ankara [number] => 639
[status] => 1 [adress] => Array ( [0] => JavaSE [1] => JavaEE [2] => GNU-Linux
[3] => PHP [4] => AngularJS [5] => Python ) )
```

Gördüğünüz gibi PHP ile JSON kullanımı oldukça basit. Kısacası hepsi bu kadar. Zaten PHP yapısı gereği oldukça basit bir programlama dili. Şayet `Freelancer` olarak çalışmayı düşünüyorsanız veyahut web için basit bir dil arıyorsanız PHP biçilmiş kaftandır. Çok kolay ve hızlı bir şekilde PHP öğrenebilirsiniz.

Son olarak JSON kullanırken oluşabilecek hata durumlarını yakalamak için kullanılan bir fonksiyona da değinelim.

7.1.3 json_last_error( )

JSON > PHP veya PHP > JSON dönüşümleri sırasında herhangi bir hata durumu meydana gelebilir. Bariz bir hata yapmışsanız zaten PHP size bu hatayı gösterir lakin JSON yapısında; syntax (söz dizimi) v.s gibi bir hata durumunda size hata gösterilmez, dönüşüm işlemi de başarısız olur. Işte bu ve buna benzer durumlarda `json_last_error( )` fonksiyonu kullanılır.

json_last_error( ) fonksiyonu bir değer döndürür. Hatanın kaynağını öğrenebilmek için bu fonksiyonun dönüş değerinin kontrol edilmesi gerekmektedir. Bu fonksiyonun ürettiği hata çıktıları aşağıdaki gibidir.

JSON_ERROR_NONE	Hata bulunamadı
JSON_ERROR_CTRL_CHAR	Kontrol karakteri hatası
JSON_ERROR_DEPTH	Yığın taşması oluştu
JSON_ERROR_SYNTAX	Söz dizim hatası
JSON_ERROR_UTF8	UTF-8 karakter kodlama hatası

Üretilen bu hata kodlarının kontrol edilmesi gerekmektedir. Aslında ne demek istediğimi aşağıdaki örneğimizi incelediğinizde daha iyi anlayacağınıza eminim.

Örneğin elimizde; UTF-8 karakter kodlamasına uymayan bir karakter kümesi olsun. Bu karakter kümesini JSON formatına çevirmeye çalışalım. Oluşabilecek hatayı kontrol edelim. Örneğimiz aşağıdaki gibi olacaktır.

```
<?php

$code="\xB1\x31";

$phpToJSON  = json_encode($code);
$error = json_last_error();
switch ($error) {
       case JSON_ERROR_NONE:
           echo "Hata bulunamadı!";
       break;
       case JSON_ERROR_DEPTH:
           echo "Yığın taşması yaşandı!";
       break;
       case JSON_ERROR_CTRL_CHAR:
           echo "Kontrol karakter hatası!";
       break;
       case JSON_ERROR_SYNTAX:
           echo "Sözdizim hatası!";
       break;
       case JSON_ERROR_UTF8:
           echo "UTF-8 karakter kodlama hatası!";
       break;
       default:
           echo "Bilinmeyen bir hata meydana geldi!";
       break;
   }
?>
```

Yukarıdaki örneğimizi inceleyelim. UTF-8 karakter kodlamasına uymayan bir karakter kümesini json_encode ( ) fonksiyonu ile JSON formatına çevirmeye çalıştık. Daha sonra oluşabilecek hatayı json_last_error ( ) fonksiyonu döndüreceği için kontrol ettik. Burada $error değişkenine, json_last_error ( ) fonksiyonununda dönen değerin atamasını yaptık. Ayrıca bir dip not düşelim. json_last_error ( ) fonksiyonundan dönen değer bir tamsayıdır.

Yukarıdaki örneğimizi çalıştırdığımızda aşağıdaki çıktıyı verecektir.

```
UTF-8 karakter kodlama hatası!
```

json_last_error ( ) fonksiyonunun kullanımına dikkat edilmelidir. Kontrol edilmezse; hata meydana gelir lakin çözümlemesi yapılamaz. O yüzden kullanırken ekstra olarak bir kontrol mekanizması yazmaya özen gösterin. Bu kontrol mekanizması; if veya switch...case olur farketmez, tamamen size kalmıştır.

KAYNAKLAR

http://www.json.org/
http://www.w3schools.com/json/default.asp
http://php.net/manual/tr/book.json.php
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON
http://www.tutorialspoint.com/json/