Fetch API

Fetch API, kaynakları (ağ genelinde dahil) getirilmesine yönelik basit bir arayüzdür. Ajax yerine alternatif olarak asenkron veri alıp ve veri göndermemizi sağlar. Promise tabanlı olduğu için async bir yapıdır. Promise ES6 ile gelen bir özelliktir. Callback fonksiyonlarının yerine kullanabiliyoruz. Fetch API'da promise yapısını kullanarak bizim Ajax'ın yerine kullanabileceğimiz bir yapı sunuyor. Ayrıca Fetch API window objesi ile birlikte geliyor. Eğer window objesini konsolda yazdırırsak bu window objesinin altında fetch metodunu görebilirsiniz.
Fetch API ile yapabildiğimiz şey bir veri sunucudan geldikten sonra işlem yapabilmektir. Peki bu ne demektir? Diyelim ki bir yere gittiniz ve sipariş veriyorsunuz. Kasadaki kişi siparişi aldıktan sonra kasada beklersek bir sonraki kişinin siparişinin vermesini engelleriz. Belki de bir sonraki kişinin siparişi çok daha hızlı hazırlanabilirdi. Bu durumda şunu da unutmamak lazım biz ne zaman sipariş ettiğimiz ürünleri gidip almak isteriz? Tabii ki siparişimiz hazırlandıktan sonra. Burada aslında yeni gelen kavramlardan bir tanesi olan Promise'i kullanırız.

Fetch Interfaces

*fetch(), bu yöntem bir kaynak almak için kullanılır.
*Headers, response/request başlıklarını temsil ederek bunları sorgulamanıza ve sonuçlara bağlı olarak farklı eylemler gerçekleştirmenize olanak tanır.
*Request , bir kaynak talebini temsil eder.
*Response, bir isteğe verilen yanıtı temsil eder.

Temel Fetch Kullanımı

Fetch API' yi kullanmak için fetch metoduna istek yapacağımız url' i parametre olarak vermek gerekiyor.

```
fetch(url);
```

fetch() metodundan sonra, metodun sonuna then() promise metodunu ekleriz:

```
.then(function() {
})
```

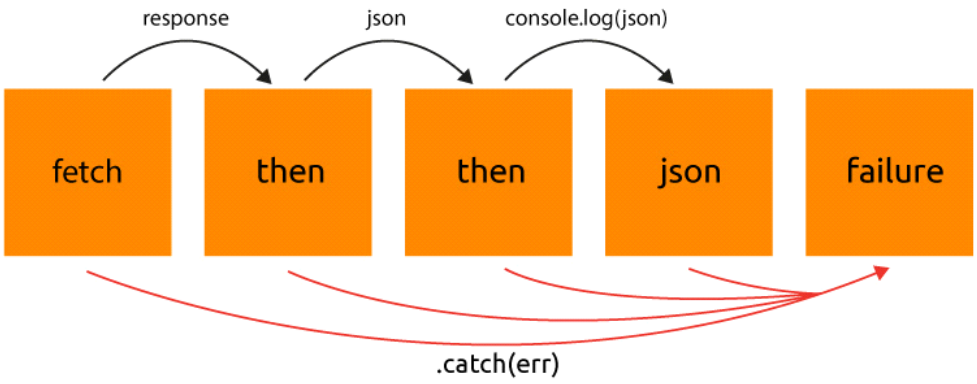
Örneğin "https://jsonplaceholder.typicode.com/todos" adresine get isteğinde bulunalım.

```
//// GET İsteği
fetch("https://jsonplaceholder.typicode.com/todos")
  .then((response) => response.json()) //parse json data
  .then(function (todos) {
    todos.forEach((todo) => {
      console.log(todo.title); //Başlıkları console' a yazdırma
    });
  });
```

POST İsteği Yapma

```
💬 POST isteği ile verimizi servera gönderelim
let payload = {
    title: "Blog Title",
    body: "lorem ipsum",
    userId:1
  }
  fetch('https://jsonplaceholder.typicode.com/posts', {
    method: "POST",
    body: JSON.stringify(payload),
    headers: {"Content-type": "application/json; charset=UTF-8"}
  })
  .then(response => response.json())
  .then(json => console.log(json))
  .catch(err => console.log(err));
```

Aşama aşama fetch().then().then().catch() yapısı:



Peki bu kod bloğu nasıl çalışır? Şimdi aşama aşama inceleyelim:

```
fetch("https://jsonplaceholder.typicode.com/todos");
```

Fetch isteği yaptığımızda then fonksiyonunu çağırır.

```
.then((response) => response.json())
```

Her şey yolunda giderse gelen veri, then içerisinde bize response olarak gelir. Ve response'u parametre olarak alırız, json içeriğini elde etmek için .json() metodunu kullanırız.

```
.then(function(todos){
    todos.forEach(todo => {
        console.log(todo.title);//Başlıkları console' a yazdırma
    });
})
```

Bu kez `then()` metodu içerisinde, önceki metottan gelen sonucu alarak console'da gösterecek bir fonksiyon yazdık. Bu fonksiyon, `todos` adında bir parametre alıyor ve her `todos` 'dan `forEach` metodu yardımıyla `todos.title` değerini alıp console'a yazdırıyoruz.

```
.catch((err) => console.log(err))
```

Ve en sonunda catch() metoduyla herhangi bir aşamada hata oluşursa, bu hatayı parametre olarak alıp console'a yazdırıyoruz.