Değişkenler

Çoğu zaman JavaScript uygulamaları bilgi ile çalışır. Örnek vermek gerekirse:

1. Online Mağaza – satılacak ürün bilgileri.
2. Sohbet Uygulaması – Kullanıcı ve mesaj bilgisi.

Değişkenler bu bilgileri tutmak için kullanılırlar.

1. Değişken

Değişken "isimlendirilmiş hafıza" olarak adlandırılır. Değişkenler ile kullanıcıları, ürünleri ve diğer tipteki bilgileri tutabiliriz.

JavaScript dilinde değişken let kelimesiyle üretilir.

Aşağıdaki cümle "mesaj" isminde bir değişken üretir (diğer bir deyişle *tanımlar*):

```
let mesaj;
```

Artık bunun içerisine bilgi koyulabilir. Bunu atama operatörü ile = yapabilirsiniz.

```
let mesaj;

mesaj = 'Merhaba'; // Merhaba karakter dizisini mesaja atadınız
```

Artık karakter dizi değişken ile ilintili olan hafıza bölgesine kaydedildi. Artık değişken ismi kullanarak bu değere erişilebilir.

```
let mesaj;

mesaj = 'Hello!';

alert(mesaj); // değişkenin içeriğini gösterir.
```

Daha açıklayıcı olması açısından değişkeni aynı anda tanımlayıp değer atayabilirsiniz.

```
let mesaj = 'Merhaba!'; // Değişken tanımlandı ve değer atandı

alert(mesaj); // Merhaba!
```

Birden fazla değişkeni bir satırda tanımlamak da mümkündür.

```
let kullanici = 'Ahmet', yas = 25, mesaj = 'Merhaba';
```

Kısa görünse bile yukarıdaki yazım önerilmez. Okunabilirlik açısından her değişkenin bir ayrı bir satırda yazılması daha iyi olacaktır.

Tabi her değişken için ayrı satır kullanırsanız biraz uzun olur fakat okunması açısından daha kolaydır.

```
let kullanici = 'Ahmet';

let yas = 25;

let mesaj = 'Merhaba';

Bazı programcılar ise şu şekilde kullanmaktadırlar:

let kullanici = 'Ahmet',
```

```
  age = 25,
  mesaj = 'Merhaba';
```

Hatta bazıları şu şekilde kullanır:

```
let kullanici = 'Ahmet'
  , yas = 25
  , mesaj = 'Merhaba';
```

Teknik olarak bu yazımların hepsi doğrudur. Gerisi sizin yazım tarzınıza kalmış. Her yiğidin yoğurt yiyişi farklıdır.

let yerine var kullanma

Eski kodlarda let yerine var kullanıldığını görürsünüz.

```
var mesaj = 'Merhaba';
```

var kelimesi *neredeyse* let ile aynı anlama gelmektedir. var kelimesi de değişken üretmeye yarar fakat bunu eski tarzda yapar.

let ile var arasında nüans farkı vardır. İkisi de istediğimizi yerine getirir. Bunun ile ilgili detaylı bilgi Eski tip "var" konusuna gelindiğinde verilecektir.

Gerçek hayat ile benzeşim

"Değişken" konsepti "kutu" olarak tanımlanabilir. Her kutunun üzerinde farklı bir etiket yapıştırılmıştır.

Örneğin mesaj değişkeni üzerinde "mesaj" yazısı olan ve değeri "Merhaba!" olan bir kutu olarak hayal edilebilir.

Kutuya istediğiniz değeri koyabilirsiniz. Ayrıca içerisindeki değeri istediğiniz kadar değiştirebilirsiniz.

```
let mesaj;
mesaj = 'Merhaba!';
mesaj = 'Dünya!'; // değer değişti
alert(mesaj);
```

Değer değiştiğinde, değişkenin eski değeri silinir.

Ayrıca iki değişken tanımlayıp içerilerindeki değerleri bir diğerine aktarabilirsiniz.

```
let merhaba = 'Merhaba Dünya!';
let mesaj;
// merhaba değişkeninin içeriğini mesaj değişkenine aktarın
mesaj = merhaba;
// artık iki değişken de aynı değeri taşır.
alert(merhaba); // Merhaba Dünya!
```

```
alert(mesaj); // Merhaba Dünya!
```

Fonksiyonel Diller

İlginç bir bilgi olarak Scala veya Erlang gibi fonksiyonel diller değişkenin değerinin değiştirilmesine izin vermez.

Böyle dillerde değer bir kutunun içerisinde sonsuza kadar saklanır. Eğer farklı bir değer kaydetmek istenirse bu diller bizi yeni bir kutu oluşturmaya iter. Eskisi yeniden kullanıp değeri değiştirilemez.

İlk başta biraz garip gelse de, bu diller genel olarak ciddi işlerin yapılabildiği dillerdir. Değişkenlerin tekrar atanamaması kodların paralel bir şekilde çalışmasında oldukça etkin öneme sahiptir. Bu diller üzerine çalışmanız programlamaya farklı açılardan bakmanızı sağlar.

Değişken isimlendirme

JavaScript dilinde değişken oluştururken iki sınırlama vardır.

1. Değişken ismi sadece harfler, rakamlar, $ ve _ karakterlerinden oluşabilir.
2. İlk karakter rakam olamaz.

Geçerli birkaç örnek şu şekildedir:

```
let kullaniciAdi;

let test123;
```

Eğer isim birden fazla kelime kullanıyorsa deveHörgücü veya camelCase küçük harfle başlanıp her kelimenin baş harfi büyük olacak şekilde devam etme yöntemine deveHörgücü yöntemi denir. Bu yöntem yaygın bir şekilde kullanılır. Örneğin : benimUzunDegiskenim gibi.

'$' işareti ve '_' işareti de isimlerde harf gibi kullanılabilir. Farklı bir anlamı yoktur.

Aşağıdaki isimlendirmeler geçerlidir:

```
let $ = 1; //  "$" adında bir değişken üret ve değerini 1 yap.

let _ = 2; // "_" adında bir değişken üret ve değerini 2 yap.

alert($ + _); // 3
```

Geçersiz isimlendirmeler:

```
let 1a; // Rakam ile başlanılmaz.

let my-name; // isimlendirmede '-' karakteri kullanılamaz.
```

Büyük küçük fark önemli

elma ve Elma iki farklı değişken tanımlar. Bu değişkenler birbirlerinden farklıdır.

İngilizce harici harfler geçerlidir fakat önerilmez.

Herhangi bir dili kullanmak mümkündür. Hatta aşağıdaki gibi resim yazısı bile kullanabilirsiniz:

```
let имя = '...';

let 我 = '...';
```

Teknik olarak bir hata olmamasına ve bu şekilde kullanıma izin verilesine rağmen genel olarak uluslararası gelenek olarak değişkenler İngilizce isimlendirilirler. En basit bir kod parçasının bile önünde uzun bir hayat olabilir. Diğer ülkedeki insanların bu kodları okuması gerekebilir.

Saklı tutulan isimler

Değişken olarak kullanılamayacak dilin kendisine saklı tuttuğu isimler mevcuttur.

Örneğin : let, class, return, function gibi kelimeleri değişken ismi olarak adlandıramazsınız.

Aşağıdaki örnek yazım yanlışı (syntax error) verecektir:

```
let let = 5; // değişken ismi "let" verilemez, hata!

let return = 5; // değişken ismi return verilemez, hata!
```

use strict kullanmadan değer atama

Normalde değişkeni kullanmak için önce tanımlamanız gerekmektedir. Fakat eski zamanlarda tanımlamadan da , let kullanmadan da , değer atamak mümkündü. Eğer use strict kullanmıyorsanız hâlâ eskisi gibi kullanabilirsiniz. Bu davranış eski kodlarla uyumluluk açısından olduğu gibi bırakılmıştır.

```
// not: bu örnekte "use strict" kullanılmamıştır

num = 5; // eğer "num" değişkeni daha önce yaratılmadıysa yaratılır ve 5 değeri atanır.

alert(num); // 5

Bu kötü bir kullanımdır. Eğer sıkı moda geçerseniz hata alırsınız.

"use strict";

num = 5; // error: num tanımlanmadı.
```

2. Sabitler

Sabit(değişmeyen) tanımlamak için let yerine const kullanabilirsiniz.

```
const benimDogumGunum = '18.04.1982';
```

const ile tanımlanan değişkenler “constants” (Sabit) olarak tanımlanır. Bunlar değiştirilemezler, değiştirilmek istendiğinde hata alınır:

```
const benimDogumGunum = '18.04.1982';

benimDogumGunum = '01.01.2001'; // hata, sabit'in değeri değiştirilemez!
```

Programcı değişkenin değerinin değişmeyeceğine eminse const bunu garantiler. Ayrıca bu kodu kullanan herkese bunun garantilendiğini bildirmiş olur.

Sabitlerin Büyük Harf İle İsimlendirilmesi

Genel kullanımda sabitler büyük harf ile isimlendirilirler. Eğer birden fazla kelimeden oluşuyorsa “_” ile bu kelimeleri ayırmak mümkündür.

Örneğin:

```
const RENK_KIRMIZI = "#F00";

const RENK_YESIL = "#0F0";

const RENK_MAVI = "#00F";

const RENK_TURUNCU = "#FF7F00";

// ...Resim seçmek istediğimizde

let renk = RENK_TURUNCU;

alert(renk); // #FF7F00
```

Yararları:

- RENK_TURUNCU "#FF7F00" a göre hatırlanması daha kolaydır.
- "#FF7F00" yazarken yanlış yazma olasılığı RENK_TURUNCU'ya göre yüksektir.
- Kodu okurken RENK_TURUNCU #FF7F00'dan daha fazla anlam ifade eder.

Sabitler için ne zaman büyük harf kullanılmalı ne zaman kullanılmamalı ?

“Sabit” değeri hiç değişmeyen demek. Fakat bazı değişkenler örneğin kırmızının hexadecimal karşılığı çalışmadan önce bilinirken bazıları çalışma zamanında hesaplanır fakat sonrasında değişmez.

Örneğin

```
const sayfaYuklenmeSuresi = /* Sayfanın yüklenme süresini tutar. */;
```

sayfaYuklenmeSuresi çalışmadan önce değeri bilinmeyen bir değerdir. Bundan dolayı normal isimlendirme kullanılır. Çalıştıktan sonra sadece bir defa tanımlanıp daha da değişmeyen bir değer olduğundan hâlâ “sabit” denilebilir.

Diğer bir deyişle büyük harfle yazılan değişken isimleri sadece önceden bilinen değerleri tanımlamak için kullanılır.

İsimlendirmeyi doğru yapmak

İsimlendirmeden konuşuyorsak düzgün isimlendirmeyi atlamamak gereklidir. Aslında en önemli konu da budur. Eğer değişken için isim bulamıyorsanız lütfen biraz daha düşünüp mantıklı bir isim bulun.

Proje daha karmaşıklaştıkça isimlendirmenin önemi daha da anlaşılır. Değişken isimlerine göre kodun yeni kodlamaya başlayan birisi tarafından mı yoksa tecrübeli birisi tarafından mı yazıldığını anlaşılabilir.

Çoğu projede zaman var olan kodların değiştirilmesi, bu kodlardan yeni fonksiyonlar yapılması üzerinedir. Yeni bir şey yapılacağında çoğunlukla eskisinin üzerine yapılır. Eski kodlara baktığınızda değişkenlere bakarak konuyu anlamak daha kolay olacaktır.

Lütfen değişkenleri isimlendirirken iyice düşünün sonrasında çok işinize yarayacak.

Birkaç kural şu şekildedir:

- İnsan-okuyabilir değişken ismi verin kullaniciAdi veya alisverisSepeti gibi.
- a, b, c gibi kısaltmaları kullanmayın. Tabi ne yaptığınızı kesin olarak biliyorsanız kullanabilirsiniz.
- İsimlerin açıklayıcı olmasına önem verin. Örneğin veri ve deger adindaki değişkenler bir şey ifade etmezler. Tabi eğer kod bloğunda bunların bir anlamı var ise kullanılabilir.
- Bazı tanımları kafanızda takımınızın kullandığı şekil ile uyumlu şekilde oturtun. Örneğin siteyi ziyaret eden kişi kullanici ise kullanıcı ile olan değişkenleri anlikKullanici veya yeniKullanici gibi kullanın fakat yeniZiyaretci veya yeniCocuk gibi kullanmayın.

Basit değil mi? Gerçekten öyle, fakat pratikte bu kadar da basit değil. Umarım bunu siz gerçekleştirirsiniz.

Tekrar mı kullanmalı yoksa yeni mi oluşturmalı?

Son olarak. Bazı miskin programcılar yeniden değişken yaratmaktansa eskisini kullanmayı yeğlerler.

Sonuç olarak değişken bir kutu gibidir üstüne yapıştırdığınız etiketi değiştirmeden içerisine farklı şeyler atılabilir. Fakat sonunda kutunun içinde ne olduğunu anlamak için tekrar tekrar kontrol etmek gerekir.

Böyle programcılar tanımlarken biraz zaman kazanırlar fakat bunun 10 mislini kodu takip etmek için harcarlar.

Fazladan bir değişken düşman değildir.

Modern JavaScript sıkıştırıcılar ve tarayıcılar kodları oldukça iyi optimize etmektedirler. Hatta farklı değerler için farklı değişken isimleri kullanmak JavaScript motorunun optimize etmesine yardımcı bile olabilir.

Özet

Verileri saklamak için değişken tanımlayabilirsiniz. Bu işlemi var veya let veya const ile yapabilirsiniz.

- let – modern değişken tanımlama. Chrome üzerinde let ile değişken tanımlamak istiyorsanız sıkı modda (strict mode) çalışmanız gerekmekte.
- var – eski tip değişken tanımlama. Yeni kodlarda çok nadir kullanılır, belki yakında hiç kullanılmayacak. Bu konu ileride let ile var arasındaki nüans farkı Eski tip "var" bölümünde incelenecek.
- const – bu da let gibi fakat değeri değiştirilemez.

Değişkenler bulundukları yerdeki anlamlarına göre isimlendirilmelidirler.

Görevler

Değişkenler ile çalışma

önem: 1

1. yonetici ve isim adında iki değişken tanımlayın.
2. "isim" değişkenine "Ahmet" atayın
3. Değeri isim den yonetici kopyalayın.
4. yonetici değişkeninin içeriğini uyarı fonksiyonuyla gösterin. (Çıktısı “Ahmet” olmalı)

Doğru isimlendirmeyi yapabilmek

önem: 2

1. Gezegenimizin isminin tutulacağı bir değişken oluşturun. Bu değişkeni nasıl isimlendirirsiniz?
2. Şu anda web sitesini ziyaret eden kişinin adının tutulduğu bir değişken oluşturun. Bu değişkeni nasıl isimlendirirsiniz?

Büyükharf ile sabit (const) kullanımı

önem: 3

Aşağıdaki koda bir göz atın:

```
const dogumGunu = '18.04.1982';

const yas = someCode(dogumGunu);
```

Gördüğünüz gibi dogumGunu adında bir tarih sabiti ve yaş adında dogumGunu değişkeninden hesaplanan bir değişken bulunmakta. (Örneğin kısa olması açısından someCode fonksiyonu tamamlanmamıştır.)

Sizce dogumGunu tamamı büyük harf olacak şekilde mi olmalı? yoksa yaş değişkeni mi büyük olmalı? Veya her ikisi de mi büyük harf olmalı?

```
const DOGUMGUNU = '18.04.1982'; // büyük harf mi olmalı?

const YAS = someCode(DOGUMGUNU); // büyük harf mi olmalı?
```