**1-PROGRAMLAMANIN TEMELLERİ:** Bilgisayar sistemleri veriyi alıp saklayan,işleyen daha sonra istenildiğinde bilgisayardan o veriyi alıp kullanabildiğimiz sistemdir yani input ve output şeklindedir.Bu sistem, donanım ve yazılım olarak iki ana branşa ayrılır.Donanım; temelinde RAM,disk ve işlemci bulunduran yapıdır.Yazılım; program, dökümanlar, protokoller vb. hepsini kapsar.

**2-JAVASCRIPTE GİRİŞ:** Günümüzde tarayıcılara çok veri akışı sağlandığı için Javascriptin dinamik yapısı onu bir çok yazılım dilinden daha popüler kılmıştır.

- ★ **RENDERING ENGINES** => chrome-Blink, int.explorer-Trident, Firefox-Gecko, Edge-EdgeHTML, opera-Blink, Safari-Webkit tarayıcı motorlarını kullanmaktadır. opera ve chrome aynı dikkat!
- ★ **JAVASCRIPT ENGINES** => chrome ve opera-V8, iexpl. ve edge-Chakra, firefox-SpiderMonkey, Safari-Nitro programlarını kullanıyor.
- ★ -Altın Örümcek ödül alan siteleri gösteriyor.Klasik sitelerde %10 gibi js kullanılır.

**3-JAVASCRIPTTE STANDARTLAR:** Tarayıcılar üzerinde çalışan hala en önemli dildir.

**-TARİHÇE** 1993-Mosaic, 1994-NetscapeNavigator, 1995-Javascript, 1997-adı ECMAScript olarak anılıyor,1998-ES2, 1999-ES3, 2006-AJAX&Jquery populer oluyor,2008-GoogleChrome çıkıyor, 2009-ES5, 20015-ES6, 2017-en popular dil

**4-TARAYICI VE JAVASCRİPT İLİŞKİSİ:** jsyi inline olarak eventlar ile ya da body içinde <script> ile ya da ayrı sayfa olarak yazabiliriz.

**5-TARAYICI KONSOLU VE ÇIKTI ALMA:** console.log() kodumuzun gidişatını kontrol etmek için kullanırız. alert ile minik bir pencere içinde çıktıyı görebiliriz.Bütün hmtl elementlerinin sahip olduğu .innerHtml etiketi ile document.querySelector(“body”).innerHtml gibi kullanabiliriz.

**6-DEĞİŞKEN VE SABİTLER:** Programlamanın temel yapı taşı değişkenlerdir.Değişkenlere variable, sabitlere constant, nesne yönelimli çalışıyorsak özelliklerine attribute denir.

- ★ **VAR** ve **LET** değişken atar, **CONST** sabit atar.
- ★ ” = “ atama operatorudur. Değişken ya da sabite değer atamak için kullanılır. Eğer var yada let kullanmadan ad=”Busra” atarsak bu default olarak var’dır.

  *let name; name = “busra” name =”feyza”*
- ★ Atanan değişkenler daha sonra güncellenebilir.
- ★ **CONST** ile sabit ataması yapılır. Sabitler daha sonra güncellenemez.
- ★ “ + ” operatörü ile metinleri birleştirebiliriz.

**7-LET, VAR VE CONST ANAHTARLARI:** VSCodeun debug özelliği sayesinde anlık olarak kodumuzdaki hataları takip edebiliriz.var ile değişken deklarasyonunu(değer atama) iki kere yapabiliriz.(aynı isme iki kere değişken atamak) ama let ile bunu yapamayız.const ile zaten yapılmaz.Var ve let de tekrar değer aktarımı yapılabilir ama const ile yapılamaz. *let name = "busra" let name ="belkiz"* hata verir ama *name = "feyza"* vermez.

★ **-Fonksiyon blok grubu -if, while, köşeli parantezler gibi diğer kalanlar** var ile blok içerisinde atama yapsak bile bu değere erişebiliriz.Ancak const ve let ile oluşturulan değerler bir scope oluşturur (yani kendilerine bir alan) ve o blok haricinde bu değerlere erişilmez.Ancak bir istisna olarak functional blockta oluşturulan var değişkeni de dışardan erişilmez.

**8-VERİ TİPLERİ:** Tüm programlama dillerinin ham maddesi veri tipleridir.(mozilla.mdn kaynak site)Jsde unicode karakter kümesi kullanılır ve bu yüzden değişken ismini türkçe atayabiliriz, diğer dillerde bu yok.

**STRING:** Parantez içinde olan değerler string olarak algılanır.

*let isim = "Busra";*

**NUMBER:** Sayısal veri tipleridir.

*let yas = 23;*

*let maas = 3550.25;*

**BOOLEAN:** Mantıksal veri tipleridir.

let evlimi = true;

**UNDEFINED:** Hem bir değer hem de bir veri tipidir. Mesela;

*let iseBaslamaTarihi;* <= burada bu değişkene bir değer atanasıya kadar bunun değeri undefineddır console.log yapınca undefined yazar ve aynı zamanda veri tipi de undefineddır.

**NULL:** Program esnasında bazen boş veri tanımlamak aktarmak istersek "" gibi yazılabilir ama null diye de kullanabiliriz. Bu değeri rezerve etmiş.oluyoruz. Bu değişkene yeni aktarım yapasıya kadar null döner, tipi de nulldur.

*let departman = null;*

➔ undefined ve nulla değer aktarımı yapılınca o andan itibaren tipleri de değişir.

➔ Bu noktaya kadar anlatılanlar **PRIMITIVE VERİ TİPLERDİR**(symbols de buraya dahil).<u>Primitiveler içinde sadece tek bir(1tane) bilgi kırıntısı saklar</u>

**9-TİP DÖNÜŞÜMLERİ:** Veri tiplerinin sonradan değişmesidir.implicit ve explicit olarak yapılır.**Implicit** dolaylı dönüşüm(js kendi kendine dönüştürür), **Explicit** bizim isteyerek dönüştürmemizdir.

**EXPLICIT DÖNÜŞÜMLER;**

★ Stringi number yapmak için **Number()**, **parseInt()**, **parseFloat()** kullanabiliriz.

*let isim = "Busra"*

*let isimSayi = Number(isim)* //NaN -not a number demektir-

➔ Js bunu sayı değil ama sayı tipi olarak algılıyor.

➔ string içinde sayısal değer olursa direkt numbera çevirir.Hem sayı hem metin olursa yine NaN döner.

➔ Number("") boş karakterler 0 olarak döner.

➔ parseInt içinde metin ve sayı varsa verilen string içindeki ilk sayıyı numbera çevirir.ParseInt() ondalık sayı alırsa veri tipi değiştirmez ancak ondalık kısmı silip integera çevirir.

➔ parseFloat() ise eğer içine string ifade alırsa yine onu number yapar ve içinde ondalık sayı varsa da onun tipini ondalıklı olarak numbera çevirir.

★ Numberı string yapmak için **String()** kullanabiliriz.

*let maas = 23465; //number*

*let maasMetin = string(maas) //string*

★ Booleanı string yapmak için **String()** kullanabiliriz.

*let evliMi = true; //boolean*

*let evliMiMetin = string(evliMiMetin) // "true" çıktısı yani string*

★ Boolean yapmak için **Boolean()** kullanabiliriz.(ya true ya false olacak)

*let sayi = 5; //number*

*let sayiBool = Boolean(sayi) //true*

★ (**FALSY DEĞERLER**)0, null, undefined, NaN, "" dışındaki değerleri booleana dönüştürmek true döndürür. Buradaki değerler false döner. Negatif sayılarda buna dahildir. Yani negatif görünce false düşünebiliriz ama 0 harici truedur.!!!

**IMPLICIT DÖNÜŞÜMLER;**

★ **+ OPERATORU** Eğer string bir veri + operatorunun sağında veya solunda varsa birleştirici operatordur ama sayısal veriler varsa toplama yapar.

★ *let metin = 25 + "";* //string

*let metin2 = 25 + Number("")* //25 + 0 = 25 number

★ **If(){}:** Parametre olarak verilen değeri implicit olarak true yada false yapar.

★ **! :** Tersine çevirir.Önündeki değer true/false ise tersi yapar ama diğer veri tiplerini önce booleana çevirir sonra ona göre tersi yapar.

**10-OPERATÖRLER:** **ATAMA OPERATÖRLERİ** : =, += , -=, *=, /= , %=

**ARİTMETİK OPERATÖRLER** : +(sayısal veri olduğunda), -, * ,/ , %(mod alma(kalan)),++(increment) değeri 1 arttıran operatör, --(decrement) değeri 1 eksilten değer

**METİN OPERATÖRLERİ** : +(birleştirici olduğunda)

★ **typeof()** sağına girilen değerin veri tipini döner.(özel operatör)Girilen değeri string olarak döner.”number”, “boolean” gibi.

**11-KARŞILAŞTIRMA OPERATÖRLERİ:** Bir karşılaştırma yaptıktan sonra geriye boolean dönen operatörlerdir. <,>, >=, >=, ==, !=, ===, !== boolean değer döndürürler.

- ➔ typeof(a<b) BOOLEAN değer döndürür.
- ➔ === hem değerleri hem de tipleri eşit midir? olarak iki kontrol yapar. 5 === “5” false döner.
- ➔ == sadece değer kontrolü yapar.Tip kontrolü yapmaz. 5 == “5” true döner.
- ➔ != bu da tip kontrolü yapmaz sadece değere bakar. 5 != “5” false(eşitler ama eşit değil? kontrolü yaptı o yüzden false döner)
- ➔ !== bu da tip ve değer kontrolü yapar. 5 !== “5” true (değerleri eşit olsa da tipleri eşit değil ve eşit değil mi kontrolü yaptığı için true döner.)

**12-MANTIKSAL OPERATÖRLER:** && and , || or , !not boolean değer dönerler.

- ➔ ! sadece sağa değer alır diğerlerinden farklı olarak. Yanındaki değerin tersini alır. !(“busra”) //false döner !(“”) // true döner.
- ➔ && sağında ve solunda değer olmalı ve her iki değer de true ise ancak o zaman true döner. Bir tanesi bile false olursa direkt false döner.
- ➔ || sağına soluna aldığı değerlerden bir tanesi bile true ise veya her ikisi de true ise true döner.Sadece ikisi de false olursa ancak o zaman false döner.

**13-KOŞULLU YAPILAR:** Tekrar tekrar çalıştırılabilen kodlardır döngü/loop denir.Kodlar derlenirken karar verip verinin işlenip işlenmeyeceğini de Koşullu Yapılar ile kodluyoruz. Bu kod akışına control flow denir.

-kodların okunaklı olması için bloklar tab boşluğu ile oluşturulur.

-köşeli parantez kullanılmazsa tek bir komut if içine dahil edilir.

- ➔ IF ELSE : if() parantez içindeki değerin koşullu bir yapı olup boolean değer dönmesi gerekir.Ve false olursa if bloğu çalışmaz.
  - else : aksi halde demektir yani if bloğunun içindeki değer çalışmazsa else içindeki çalışır.Koşul almaz. else{} yazılır.
  - else if : eğer olarak düşünebiliriz. if iiçindeki değere alternatif koşullar varsa o zaman kullanılır.
  - if yapısında özgürüz ihtiyacımız ne ve ne kadarsa o kadar kullanabiliriz.
- ➔ SWITCH : switch() parantez içine koşul almaz. Koşullar karşılaştırmalar case içinde bulunur. Ve yaptırmak istediklerimizi case içine yazarız.

  -switch bloğu içinde == kontrolü yapılır. Yani sadece değer!

  -iften farkı bir değer çalışırsa devamında case varsa onlar da çalışır.Bu davranışı engellemek için break kullanılır.

  -default koşulu da en sona yazılır ifteki else gibidir.