Odev Cevaplar

1. Javascript web tarayicilari ve sunucu tarafinda calisabilen bir programlama dilidir. Web tarayicilarinda kullanim amaci sayfalari kullanici ile etkilesimli hale getirmek ve sayfa iceriginin dinamikligini saglamaktir. Serverside calisitirilmasi ise Node.js isimli bir Runtime Environment ile gerceklestirilmektedir. Bu environment'in temel amaci javacript in browser disinda calistirilabilmesidir.

   Javascript dilinin tarihsel gelisimi ise asagidaki gibi siralanabilir.
   a. **Netscape ve Mocha (1995):** Javascript ilk olarak Netscape navigator tarayicisi icin gelistirildi. Bu donemde "Mocha" ve "Livescript" gibi isimlerle anildigi olsa da sonrasinda pazarlama hamlesi olarak isminin Java'ya benzemesi icin son ismi "Javascript" olarak belirlendi.
   b. **JScript ve Internet Explorer (1996):** Microsoft kendi internet tarayicisi Internet Explorer icin JScript adinda kendi javacsript versiyonunu olusturdu ancak bu durum tarayicilar arasinda uyumsuzluk problemleri yasanmasina sebep oldu.
   c. **ECMAScript Standartlari (1997):** Javascript in standartlastirilmasi amaciyla Europen Computer Manuifacturer's Association (ECMA) tarafindan bir komite kuruldu. Bu kurul standartlar olusturup javascript'in farkli tarayicilarda yasadigi uyumsuzluk sorununu cozdu.
   d. **DOM ve AJAX (2000):** JQuery gibi kutuphaneler Javascript gibi dilleri daha etkili ve tarayici uyumlu hale getirmek icin araclar sundu.
   e. **Node.js ve Server-Side Javascript (2009):** Javascript kodunun tarayici disinda da calistirilabilmesine olanak saglayan Node.js gelistirildi. Boylelikle gelistiricilerin bir dil ile hem tarayici hem sunu gelistirme yapabilmesinin -fullstack gelistirme- yolu acilmis oldu.
   f. **ES6 ve Sonrasi (2015-...):** ECMAScript 2015 (ES6) ile birlikte javascript dilinin diger dillerde olan ama javascript dilinda olmayan ozellikleri eklenerek dilin gucu arttirilmistir.
   g. **Gunumuz:** Gunumuzde React, Angular ve Vue gibi gelistirme kutuphaneler/frameworkler kullanilarak javascript gelistirme surecleri cok daha etkili bir sekilde yapilabiliyor.

   ECMAScript 6'dan sonra gelen ana ozellikler ise asagidaki gibi siralanabilir.
   1. **ES6 (2015):** arrows, classes, enhanced object literals, template strings, destructuring, default, rest, spread, let, const, iterators, for of, generators, unicode, modules, module loaders, map, set, weakmap, weakset, proxies, symbols, subclasses, built-ins, promises, math, number, string, array, object api, binary and octal literals, reflect api, tail calls
   2. **ES7 (2016):** array prototype, exponentiation operator
   3. **ES8 (2017):** async functions, shared memory and atomics
   4. **ES9 (2018):** async iteration, res/spread operators
2. Asagidaki gibi siralanabilir:
   a. Java derlenerek calistirilan bir programlama diliyken javascript interpret edilerek calistirilir.

    b. Java bir object oriented programlama dilidir ancak javascript sonradan eklene ozelliklerle object oriented gibi davranmasi calisilan bir dildir.
    c. Java programlama dilinde runtime sirasinda null donmesi gibi durumlarda program hata verirken javacript dilinde bu durumda degisken null veya undefined atanarak program bu degerlerle calismaya devam eder.
3. Javascripteki veri tipleri complex ve primitive olarak ikiye ayrilabilir. Primitive tipler number, string, boolean ve symbol (null&undefined) iken complex veri turleri objeler ve fonksyionlardir.
4. Null gelistirici tarafindan atanan ve ilgili degerin olmadigini gosteren bir veri tipidir. Ancak undefined runtime sirasinda ilgili degerin olmadigi ortaya ciktiginda gosterilen degerdir. Esitlik karsilastirmasinda bu iki tip deger olarak aynidir ancak tip karsilastirmasi da yapildiginda false donmektedir.
5. NaN Not-a-Number in kisaltmasidir. Kod icerisinde sayi olmayan bir degere sayilara tanimli bir islem tanimlamaya calistigimizda bu hatayi aliriz.
6. Javascriptte yorum satiri eklemek icin // veya /* */ kullanabiliriz.
7. Global degisken en dis scope'ta tanimlanmis ve her yerden erisilebilen degiskendir.
8. this anahtar kelimesi belirli bir scope icerisindeki degiskene erismek istedigimizde kullanilir. En cok kullanildigi alan class scope'laridir.
9. == iki degiskeni degerleri acisindan kiyaslamak istedigimizde kullanilir. === ise iki degiskeni hem degerler hem degisken tipleri acisindan kiyaslamak istedigimizde kullanilir.
10. .

No	const	let
1	degeri sonradan degistirilemeyen bir degisken yaratir	degeri sonradan degistirilebilir
2	degeri tanimlanirken atanmak zorundadir	Sonradan deger atamasi yapilabilir

11. Farklar asagidaki gibidir:
    - Arrow fonksiyonunun yazimi ozellikle tek satirli ve deger donen fonksiyonlarda daha kolaydir.
    - Arrow fonksiyonlar default olarak arguman almazlar. Erisilmeye calisildiginda arguments is no defined hatasi alinir.
    - Arrow fonksiyonlar default olarak this binding islemi yapmazlar.
    - Constructor fonksiyonlar arrow fonksiyon olarak tanimalanamaz.
    - Arrow fonksiyonlar declare edilemez ancak expression icerisinde tanimlanabilirler.

12. Suslu parantezler ile scope acilarak tanimlanabilir.
13. Pure fonksiyon ayni degerler parametre olarak verildigi her seferinde ayni sonucu donduren fonksiyonlara denir.
14. Rest operatoru temel olarak fonksiyonlara sinirsiz sayida eleman alirken kullanilir. Girilen birden fazla degeri bir array icerisinde toplayarak doner.

15. Object destructuring spread operatoru ile bir obje veya array icerisindeki degerleri teker teker disari cikarip dondurur.
16. Asagidaki gibi olusturulabilir.
    a.

```
let obj = {
      a: 5,
      b: 6
}
```

    b.

```
function ObjectClass(){
      this.a = 5
      this.b = 6
}
let obj = new ObjectClass()
```

    c.

```
class ObjectClass{
      a = 5
      b = 6
}
let obj = new ObjectClass()
```

17.

```
let obj = {
      id : "123",
      school : "Bilkent"
}

let keyLengths = {}

for( let key of Object.keys(obj) ){
      keyLengths[key] = key.length
}

let valueLengths = {}
for( let [key, val] of Object.entries(obj) ){
```

```
    valueLengths[key] = val.length
}
```

18.

no	Local storage	Session storage	Cookie
1	Boyutu 10MB ile sinirlidir.	Boyutu 5MB ile sinirlidir.	Boyutu 4KB ile sinirlidir
2	Tarayici kapandiginda degeri silinmez.	Tarayici kapandiginda degeri silinir.	Belirli bir sure sonunda degerinin silinmesi icin setlenebilir.
3	Kullanicinin durum offline durumda da ihtiyaci olabilecek bilgileri icerir.	Web sitelerinin performansini arttirmak icin session bazli bilgileri iceririr.	Session id ler gibi uzun bir sure tutulmasi gereken bilgileri tutar.
4	Kullanicinin site ile ilgili local tercihlerini ve ayarlari tutmak icin kullanilir.	Credentials gibi hassas datalari tutmak icin iyidir.	Kullanici tercihleri gibi hassas olmayan datalari tutmak icin kullanilir.
5	HTML 5 ile gelmistir	HTML 5 ile gelmistir	En eski storage turudur.
6			Istek atilirken gonderilir.
7	Tarayici ve sistemde tutulur.	Yalnizca tarayicida tutulur.	Yalnizca tarayicida tutulur.

19. Senkron islemler bitene kadar bir sonraki islemin gerceklesmesine izin vermez. Asenkron islemler sonraki kodun akisina izin verirken islemlerini paralelde gerceklestirir.
20. Promise ler asenkron islemleri yonetmek icin kullanilir. Callback functionlar yerine gelistirilmesinin nedeni ise callback fonksiyonlarda coklu asenkron islem yonetiminde yonetimin zorlasmasidir.

Array Sorulari

1.

```
console.log(dolap.pop())
```

2.

```
dolap.splice(0, 1, "Hat")
console.log(dolap)
```

3.

```
let isArray = Array.isArray(dolap)
```

4.

```
let pantExist  = dolap.includes("Pant")
let pantExist2 = dolap.indexOf("Pant") !== -1
let pantExist3 = dolap.some((val)=> {
    return val === "Pants"
})
```

5.

```
let sum = dolap.reduce((totalSum, val)=> {
    return totalSum + val.length
}, 0)
```

6.

```
let dolapBig = dolap.map((val)=>{
    return val.toUpperCase()
})

let dolapBig2 = [];
for(let val of dolap) {
    dolapBig2.push(val.toUpperCase())
}

let dolapBig3 = []
dolap.forEach((val)=> {
    dolapBig3.push(val.toUpperCase())
})
```

7.

```
let obj = {...dolap}
```

8. slice bir arrayden baslangic ve bitis indexlerini verip parca almamizi saglar. Array

dondurur. splice ise baslangic indexi ve silinecek eleman sayisini parametre olarak alarak belirtilen index e verdigimiz elemanlari atama islemi yapar.

Kod ciktilari

1.
   Error 1
   Success 4

   Promise reject oldugu icin ilk catch metodu calisir. Sonrasinda problem handle edildigi icin catch icerisinde resolve edilmis bir promise doner dolayisiyla sonraki then blogu da calisir.
2.
   success
   Defeat
   error
   Error caught
   Success: test

   job(true) resolve ettigi icin ilk then bloguna girip ekrana success yazdirdik.
   Bu then blogunda yeni bir promise dondurdu job(true)
   Bu promise resolve oldugu icin sonraki then bloguna girdik
   Buradaki data onceki promise'te resolve edilen value'ya karsilik geldi. Dolayisiyla burada Defeat throw ettik.
   Hatayi throw ettigimiz icin sonraki catch bloguna girdik ve ekrana mesaji yazdirdik.
   Bu blokta Error u throw degil return ettigimiz icin rejected bir state donmedi dolayisiyla son then blok una girip islemimiz tamamlandi.