JavaScript

Javascript tarayıcı üzerinde çalıştığı için her tarayıcının kendine göre bir yorumlama şekli var.

for (x in dizi) {

console.log(x);

} x olarak indis değerini yazar

for (x of dizi) {

console.log(x);

} dizinin x. elemanını yazar

debugger; eklediğin yerde geri kalan kodları çalıştırmaz

copyWithin: dizilerde kopyalama yapıyor

entries: dizinin elemanlarında geziniyor

every: dizi elemanlarını kontrol eder

fill: array elemanlarını sabit bir değer ile değiştirir(number dışında bir şey de olabilir), 3 değer alır: değer, başlangıç ve bitiş

ECMAScript6

const: read-only

let: sadece tanımlandığı { } içerisinde geçerli oluyor, blok dışında tanımlanıp blok içinde değeri değiştirilirse blok dışındaki değeri aynı kalır.

Destructuring

var list = [1,2,3];

var [a,b,c] = list;

var a = 1;

var a = 2;

A yı b ye b yi a ya atmak için normalde şunu yapıyoruz:

Var tmp = a;

Var a = b;

Var b = a;

Es6 ile şu şekilde:

[a, b] = [b, a];

var list = [1,2,3];

var [a, ,c] = list;

burada b yi yazmayıp boş bıraktığımız için list dizisinin 0. ve 2. indisindeki elemanları a ya ve c ye atar.

var obj = {a: 1, b: 2, c: 3};

var {a, b, c} = obj;

farklı isimlerle değişkenleri almak istersek:

var {a: first, b: second, c: third} = obj;

var obj = {a: 1, b: 2, c: {id: 3, name: “Hello”}};

var {a, b, c: {name} } = obj;

var obj = {a: 1, b: 2, c: 3};

var {a, b, c = 0} = obj;

c ye default değer verdik, c için bir değer yoksa 0 yazar, yukarıdaki örnekten c = 3 tür

```
function test(obj) {
        console.log(obj.userId);
}
var obj = {userId: 45};
test(obj);
```

tüm bunların es6 ile daha kısa yazımı:

```
function test({userId}) {
        console.log(userId);
```

```
}
test( {userId: 45});
```

map ve filter:

```
var arr = [1, 2 ,3 ,4 ,5 ,6 ,7 ,8 ,9];
var odds = [];
var pairs = [];
arr.forEach(function(n) {
        if(n % 2 == 1) {
                odds.push(n);
        } else {
                pairs.push(n);
        }
})
```

Bunu es6 filter ile daha kısa yazabiliriz:

```
var odds = arr.filter(function(n) { return n%2 == 1});
var pairs = arr.filter(function(n) { return n%2 == 0});

var arr2 = arr.map(funstion(n) { return n+1} ); tüm elemanları 1 artırdı
```

arrow function

this kukanımını ortadan kaldırır, süslü paranter kullanımı azalır

spread operatörü

yayma yapar

```
var arr = [1, 2, 3, 4, 5];
var arr2 = [...arr, 6, 7];
```

objelerde de kullanılabilir

spread operatöründe sıralama önemli:

```
var obj1 = {a: 1, b: 2, c: 3};
```

var obj2 = {a: 4};

var obj3 = {…obj2, obj1}; burada obj3 = {4,2,3} olur

var obj3= {…obj2, …obj1} ; burada ise obj3 = {1, 2, 3} olur.

filter: koşulu sağlayan tüm değerleri döndürür

find: tek bir değer döndürür, yazdığımız koşula uyan ilk değeri

includes: verilen değeri içerip içermediğini kontrol eder, true-false

repeat: stringleri tekrar ettirmek için kullanılabiliyor

stratsWith: o değerle başlayıp başlamadığını kontrol eder

objenin varlığını kontrol ederken includes kullanamayız, find kullanmamız gerek

endsWith: : o değerle bitip bitmediğini kontrol eder

Semboller

unique ve ilkeldir, dışarıdan ulaşılamaz

herhangi bir değerin başına + veya – koyarsak sayıya çevirir

array ve obje unique tir, string unique değil