

JavaScript Ankara

ANKARA

JS

javascriptankara.org

Üstün Özgür
@ustunozgur
ustunozgur.com

An orange square with the text "ES6" in white, bold, sans-serif font.

ES6

ECMA-262

EcmaScript 6

Tarihçe

- JavaScript: Uygulama
- EcmaScript: Spec (Belirtim)
- Netscape JavaScript ve Microsoft JScript, ActionScript
- ECMA standard organizasyonu 262 nolu proje
- TC-39: Teknik Komite 39

Tarihçe

- ES 3: 1999
- ES 4: Terk edildi
- ES 5 : 2009 ve 5.1: 2011
 - 258 sayfa
- ES 6: Taslak tamamlandı,
 - 657 Sayfa
 - Son Hali Haziran 2015
 - Harmony (Uyum)
- ES 7: Devam ediyor

Temel Başlıklar

- Var yerine Let ve Const kelimeleri
- Fonksiyonlardaki değişiklikler
- Object'lerdeki değişiklikler ve destructuring (yapı bozum, patern eşleme)
- Class özelliği
- Taslak (template) stringler
- Promiseler ve liste işleme (comprehension)

Let ve Const

- JS, görünüm olarak C'ye benzer, ama blok kapsamlı değildir! En büyük farklardan biri
- Var fonksiyon kapsamlı (scope)
- Let ve Const Blok kapsamlı
- Const değiştirilemez

Var ve Let

Hoisting (Tepeye Çıkarma)

```
function foo () {  
  var i;  
  console.log(i); undefined  
  
  for (var i = 0; i < 10; i++) {  
    console.log(i);  
  }  
  
  console.log(i); 10  
}
```

Var ve Let

```
function foo () {
```

```
    console.log(i); // hata
```

```
    for (let i = 0; i < 10; i++) {
```

```
        console.log(i); // i sadece bu blokta tanımlı
    }
```

```
    console.log(i); // hata
}
```


Var ve Let

```
function foo () {
```

```
    console.log(i); // hata
```

```
    for (let i = 0; i < 10; i++) {
```

```
        console.log(i); // i sadece bu blokta tanımlı
    }
```

```
    console.log(i); // hata
}
```

Örnek

```
var links = document.getElementsByTagName('a')  
for (var i = 0, len = links.length; i < len; i++){  
    links[i].addEventListener('click', function(e){  
        alert('You clicked on link ' + i)  
    }, false)  
}
```

DEMO

var_problem.html

Çözüm

Değişkeni fonksiyona hapsetmek

```
var links = document.getElementsByTagName('a')
```

```
for (var i = 0, len = links.length; i < len; i++){
```

```
  (function (j) {
```

```
    links[j].addEventListener('click', function(e){
```

```
      alert('You clicked on link ' + j)
```

```
    }, false)
```

```
  })(i);
```

```
}
```

DEMO

var_cozum.html

Çözüm: Let ile

```
var links = document.getElementsByTagName('a')  
for (let i = 0, len = links.length; i < len; i++){  
    links[i].addEventListener('click', function(e){  
        alert('You clicked on link ' + i)  
    }, false)  
}
```

DEMO
let_cozum.html

Const

```
const PI = 3.14;
```

```
PI = 3; // Hata
```

Değişkenler ne kadar değişmezse o kadar iyi!

DEMO
let_const.js

Fonksiyonlardaki Değişiklikler

- Default parametreler `function foo(name="Ustun")`
- Rest parametreleri `function foo(name, ...digerleri)`
- Destructured parametreler `function foo({ad, soyad})`

Default Parametreler

```
function merhaba(ad="Ustun", selam="Merhaba") {  
    console.log(selam + " " + ad);  
}
```

```
merhaba();
```

```
merhaba("Ahmet");
```

```
merhaba("Mehmet", "Hello");
```

Keyword param değil!

Bu çalışmaz

~~merhaba(selam="Hello", ad="Ozgur")~~

DEMO

functions_default_params.js

Rest parametreleri

```
function topla(ilkDeger, ...sayilar) {
```

```
  var toplam = ilkDeger;
```

```
    for (var i = 0; i < sayilar.length; i++) {
```

```
      toplam += sayilar[i];
```

```
    }
```

```
    return toplam;
```

```
  }
```

```
  topla(15, 1, 2, 3);
```

```
  sayilar = [1, 2, 3]
```

DEMO

functions_rest_params.js

Parametre Destructuring (Yapıbozum ya da Parçalama)

```
function merhaba(ad, options) {  
    var selam;  
  
    var dil = options.dil;  
  
    if (dil == "en") selam = "Hello";  
    if (dil == "tr") selam = "Merhaba";  
  
    return selam + " " + ad;  
}
```

```
function merhaba(ad, {dil}) {  
    var selam;  
  
    if (dil == "en") selam = "Hello";  
    if (dil == "tr") selam = "Merhaba";  
  
    return selam + " " + ad;  
}
```

DEMO

functions_param_destructuring.js

```
function setCookie(name, value, options) {
```

```
    options = options || {};
```

```
    var secure = options.secure,  
        path = options.path,  
        domain = options.domain,  
        expires = options.expires;
```

```
    // ...
```

```
}
```

```
setCookie("type", "js", {  
    secure: true,  
    expires: 60000  
});
```

```
function setCookie(name, value, { secure, path, domain, expires }) {
```

```
    // ...
```

```
}
```

Spread operatörü

- `Math.max(1, 2, 3); 3`
- `Math.max([1, 2, 3]); // NaN`
- `var a = [1, 2, 3]; Math.max(sayilar); // NaN`
- `Math.max.apply(Math, sayilar); // 3`
- `Math.max(...sayilar); // 3`



Ok (Arrow) fonksiyonları

```
var bar = (a, b) => a + b;
```

```
var foo = function (a,b) {
```

```
  return a + b; }
```

```
foo(1,2) ; // 3
```

```
bar(1,2); // 3
```

- `var sayilar = [1, 2, 3, 4];`
- `sayilar.filter(x => x % 2 === 1)`
- `sayilar.reduce((a,b) => a * b)`

DEMO
arrow_functions.js

Ok fonksiyonları ve this kelimesi

```
var x = {  
  ad: "Ustun",  
  merhaba: function () {  
    var that = this;  
    var helper = function () {  
      console.log("Adım ", this.ad);  
    };  
    helper();  
  }  
};  
x.merhaba()
```

Ok Fonksiyonlarında this, tanımlandığı yerdeki this'tir (lexical scope vs dinamik scope)

```
var x = {  
  ad: "Ustun",  
  merhaba: function () {  
    var helper = () => {  
      console.log("Adım ", this.ad);  
    };  
    helper();  
  }  
};  
x.merhaba()
```

DEMO
this.js

Nesnelerdeki Değişiklikler

Destructuring (Yapıbozum ya da Ayırma ya da Katı OlmayanPatern Eşleme)

```
var ustun = {ad: "Ustun", soyad: "Ozgur"}
```

```
var ad = ustun.ad;
```

```
var soyad = ustun.soyad;
```

```
var {ad, soyad} = {ad: "Ustun", soyad: "Ozgur"}
```

```
var {ad: isim} = ustun;
```

Destructuring

- Array destructuring
- `var [a,b] = [1,2];`
- `var [a,b] = [b,a]; // Swap`
- Derrin destructuring

Object Oluştururken Kısayol

```
age = 30; name = "Ustun"; location = "Ankara";
```

```
ustun = {name: name, age: age, location: location};
```

```
age = 45; name = "Ahmet"; location = "Antalya";
```

```
ahmet = {name, age, location};
```

DEMO
objects.js

Object Oluştururken Kısayol

```
var ustun = {  
  
  ad: "Ustun",  
  
  adiniSoyle: function () {  
  
    console.log("Ben " + this.ad);  
  
  }  
  
}
```

```
var ustun = {  
  
  ad: "Ustun",  
  
  adiniSoyle() {  
  
    console.log("Ben " + this.ad);  
  
  }  
  
}
```

Objelerde Computed (Hesaplanmış) Özellik

<code>var fieldName = "firstName";</code>	<code>a.fieldName == "ustun"</code>
<code>var a = {</code>	<code>a.firstName ? undefined</code>
<code> fieldName: "ustun";</code>	<code>a[fieldName] = "ustun";</code>
<code>}</code>	<code>a.firstName == "ustun"</code>
 <code>var a = {</code>	
<code> [fieldName]: "ustun";</code>	
<code>}</code>	

Template (Taslak) Stringler

```
ad = "Ustun", yas = 30;
```

```
console.log("Adim " + ad + ".Yasim " + yas);
```

```
console.log(`Adim ${ad}.Yaşım ${yas}`);
```

```
console.log(`Bu bir
```

```
cok satirli metin`);
```

Taslak stringler

- Etiketli taslak stringler (tagged templates)
- `safe`Merhaba ${name}`;`
- `uppercase`Merhaba ${name}`;`
- `var safe = function (sabitler, ...degiskenler) { ...}`
- `var uppercase = function (sabitler, ...degiskenler) { ...}`

Class Kelimesi ve Sınıflar

- class ve extends
- constructor
- prototiplere dönüşüyor

Class Kelimesi

DEMO

Promise'ler

```
ogrenci = ogrenciyiBul(123)
```

```
sinif = sinifiBul(ogrenci)
```

```
okul = okuluBul(sinif)
```

Callbackler ve Pyramid of Doom

```
ogrenci = ogrenciyiBul(123, function (ogrenci) {  
  sinifiBul(ogrenci, function (sinif) {  
    okuluBul(sinif, function (okul) {  
      console.log(okul);  
    })  
  })  
})
```

Pyramid of Doom Çözümü

```
ogrenciyiBul(123)
```

```
.then(sinifiBul)
```

```
.then(okuluBul)
```

```
.then(function (okul) {console.log(okul);})
```

```
.catch(function () { console.log("hata oluştu")})
```

Promise

- `new Promise(resolve, reject)`
- `Promise.all([promise1, promise2]).then`
- `Promise.race([promise1, promise2]).then`

DEMO

Modüller

- `import myfunc from mylib;`
- `export myfunc;`

List Comprehensions

- `var x = [for (i of [0, 1, 2, 3]) i * i];`
- `[0, 1, 4, 9]`
- `var y = [for (i of [0, 1, 2, 3]) if (i % 2 === 0) i * i * i];`
- `[0, 8]`

Daha Fazla Bilgi

- Wiki: <http://wiki.ecmascript.org/>
- Understanding ECMAScript 6
 - <https://leanpub.com/understandings6/>
- Taslaklar: http://wiki.ecmascript.org/doku.php?id=harmony:specification_drafts
- <https://github.com/ustun/ecmascript-6-sunum>