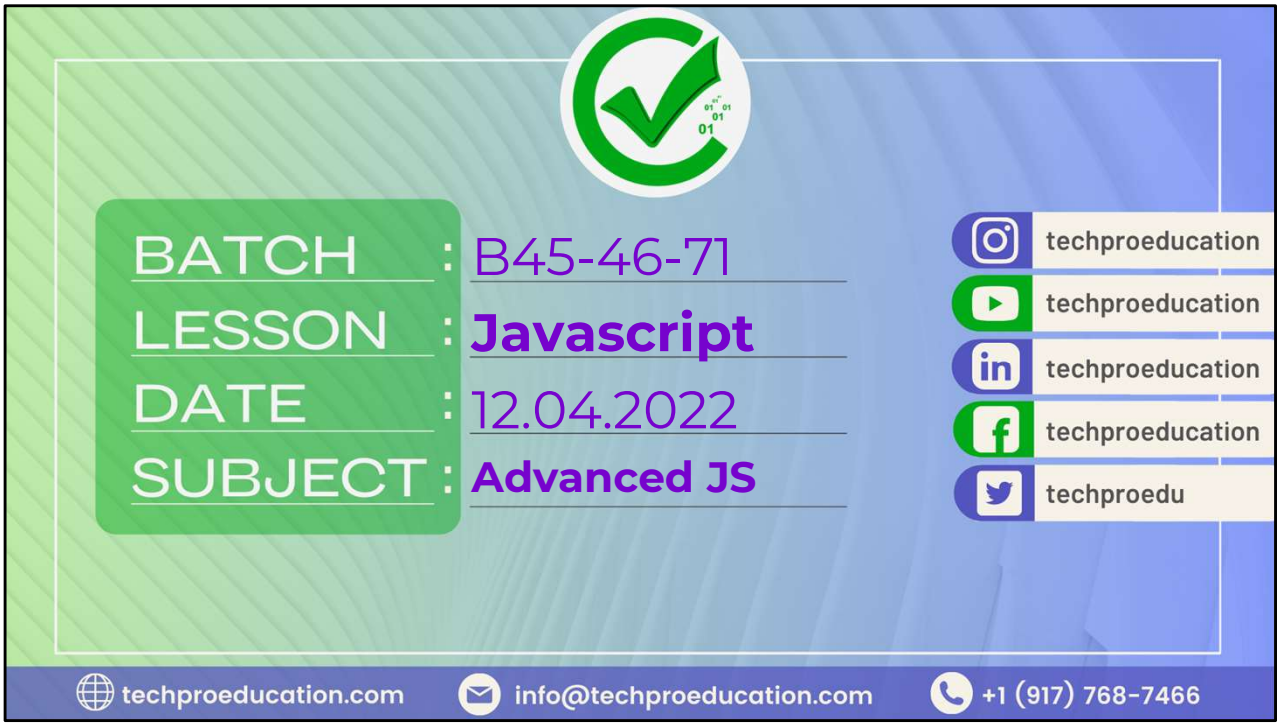
BATCH : B45-46-71
LESSON : Javascript
DATE : 12.04.2022
SUBJECT : Advanced JS
techproeducation
techproeducation
techproeducation
techproeducation
techproedu
techproeducation.com
info@techproeducation.com
+1 (917) 768-7466

Spread Operator

Örnek: Bir dizi içindeki sayıları ayrı ayrı parametre olarak topla fonksiyonuna gönderen kodları yazınız.

```
function topla(s1, s2, s3){
     return s1 + s2+ s3;
}

const sayilar = [12,56,17];

console.log(topla(...sayilar));
```

Sayılar dizisini genişletip her bir elemanını ayrı bir değişken haline getirdi.

Spread Operator

```
let numberStore = [0, 1, 2];
let newNumber = 12;
numberStore = [...numberStore, newNumber, 15];
```

numberStore dizisi başka bir dizi içinde açılarak ve yeni elemanlar eklenerek güncellendi.

Spread Operator

```
const student = {
    firstName: "Ali",
    lastName: "Gel",
    no:"234",
    phone:"23423423423"
};

let newStudent = {...student, firstName:"Veli"};
console.log(student, newStudent);
```

student nesnesini genişletip, firstName özelliğinin değeri değiştirip yeni bir nesne oluşturuyor.

Destructuring

ES 6

Bir nesnenin değerlerinin değişkenlere açılması işlemine destructuring denir.

```
let { variable1, variable2 } = object;
```

Destructuring

```
const student = {
    firstName: "Ali",
    lastName: "Gel",
    no:"234",
    phone:"23423423423«
};

const { firstName, lastName } = student;
console.log(firstName);
```

Student nesnesi içindeki, "Ali" ve "Gel" değerlerini ayrı ayrı firstName ve lastName değişkenlerine aktarmış

Destructuring

```
const student = {
    firstName: "Ali",
    lastName: "Gel",
    no:"234",
    phone:"23423423423"
};

function sayHello({firstName, lastName}){
    console.log(`Hello ${firstName} ${lastName}`);
}

sayHello(student);
```

Fonksiyon parametre olarak nesne alırken, eğer süslü parantezler içinde nesnenin key leri yazılırsa yine destructuring yapılmış olur.


Synchronous vs Asynchronous
Javscript senkron bir yapıda çalışır. Yani kodlar sırayla çalıştırılmaktadır. Ancak bazı durumlarda asenkron moda geçerek çalışma hızını artırır.
setTimeout
setInterval
fetch

Synchronous vs Asynchronous

```
function getData(){
    setTimeout( ()=> {
        console.log("ok2");
    },500);
}

console.log("ok1");
getData();
console.log("ok3");
```

ok1
ok3
ok2

Callbacks

Asenkron çalışan fonksiyonlardan veri döndürmek için callback fonksiyonlar kullanılır.

```
.
.
.
getData( (data)=> {


});
```

```
function getData(callback){
    ...
    callback("dönen data");
}
```

Callbacks

```
function getData(callback){
    setTimeout( ()=> {
        callback("ok2");
    },500);
}

console.log("ok1");
getData( (msg)=> {
    console.log(msg);
});
console.log("ok3");
```

ok1
ok3
ok2