# ECMA Script 6 JS

JS ECMA Script 6

# About me

## Vijay Shivakumar

Designer | Developer | Trainer


Training on web and Adobe products from past 10+ years

JS ECMA Script 6

# About the Program

Published on June 17, 2015

Formally "ECMA Script 2015"

6 years! from ES5

JS ECMA Script 6

# Why ES6 ?

Modern syntax fixing pitfalls

Better support for large applications

No (or few) breaking changes

JS ECMA Script 6

# Browser Support…

Modern browsers and io.js (node.js) support half of ES6 features.

Safari 9 (WebKit) will support many ES6 features soon.

Except for IE11…

Check http://www.caniuse.com

JS ECMA Script 6

# Libraries to bridge the support now…

BabelJs: https://babeljs.io/

Traceur: https://github.com/google/traceur-compiler

JS ECMA Script 6

# What's new in ES6 ?

Block Scope (let/const)
Destructuring
Default Parameters
Rest Parameters
Spread Operators
Arrow Functions ( Fat Arrow Functions)
Iterators
Generators

Template Strings
Classes
Modules
Extended Objects
Promises
Proxy
Set and Map

JS ECMA Script 6

# Block Scope in ES6

## VAR

- Globally available in the function or object where it is declared
- Hoisted
- Names can be declared many times and they override previously assigned value

## LET / CONST

- Only available in the block {} in which it is declared
- Not Hoisted
- Names can only be declared once per block

JS ECMA Script 6

# Destructuring

**Breaking down a complex structure into simpler parts.**

In JavaScript, this complex structure is usually an object or an array.

You can also handle nested structures by using nested destructuring syntax

```
var heroes = { firstHero : “Batman”, secondHero: “Superman”}
var {firstHero : fh, secondHero : sh} = heroes;
```

JS ECMA Script 6

# Generics in ES6

Generics are templates that allow the same function to work with different data types.

Good when we create reusable code that preserve the data type that go in and out of them.

You can use data type markers to denote a data type and refer those markers to represent the same data type in side a function / program

---

```
function getType<T>(value:T):string{
    return typeof (value);
};
getType(“hello there”) ; // says string
getType(true) ;// says boolean
```

JS ECMA Script 6

# Block Scope in ES6

## VAR

- Globally Available in the function which it is declared
- Hoisted
- Names can be declared many times

## LET / CONST

- Only available in the block in which it is declared
- Not Hoisted
- Names can only be declared once per block

JS ECMA Script 6

# Block Scope in ES6

JS ECMA Script 6

# Generics in ES6

Generics are templates that allow the same function to work with different data types.

Good when we create reusable code that preserve the data type that go in and out of them.

You can use data type markers to denote a data type and refer those markers to represent the same data type in side a function / program

---

```
function getType<T>(value:T):string{
    return typeof (value);
};
getType(“hello there”) ; // says string
getType(true) ;// says boolean
```

JS ECMA Script 6

# Thank you

please forward your comments and feedback

vijay.shivu@gmail.com