ES6
And
TypeScript
Overview

ES6

- Block Scope (let/const)
- Arrow Functions
- Destructuring
- Classes and Inheritance
- Modules

TypeScript

- All features of ES6
- New Features
  - Data types - number, string, boolean, etc..
  - Interface

Block Scope (let/const)

ES5 has only function scope

```
function foo() {
    var num = 1;
    // ... to many statements
    if(true_condition) {
        // same scope! overwrite above "num"!
        var num = 2;
    }
    console.log(num);   // 2
}
```

ES6 has block scope

- let and const create block scope
- no hoisting and no more ‘var’

```
function foo() {
    let num = 1;
    // ... to many statements
    if(true_condition) {
        // different scope!
        let num = 2;
    }
    console.log(num);   // 1
}
```

- const can not be reassigned
- const are immutable value (not immutable object)

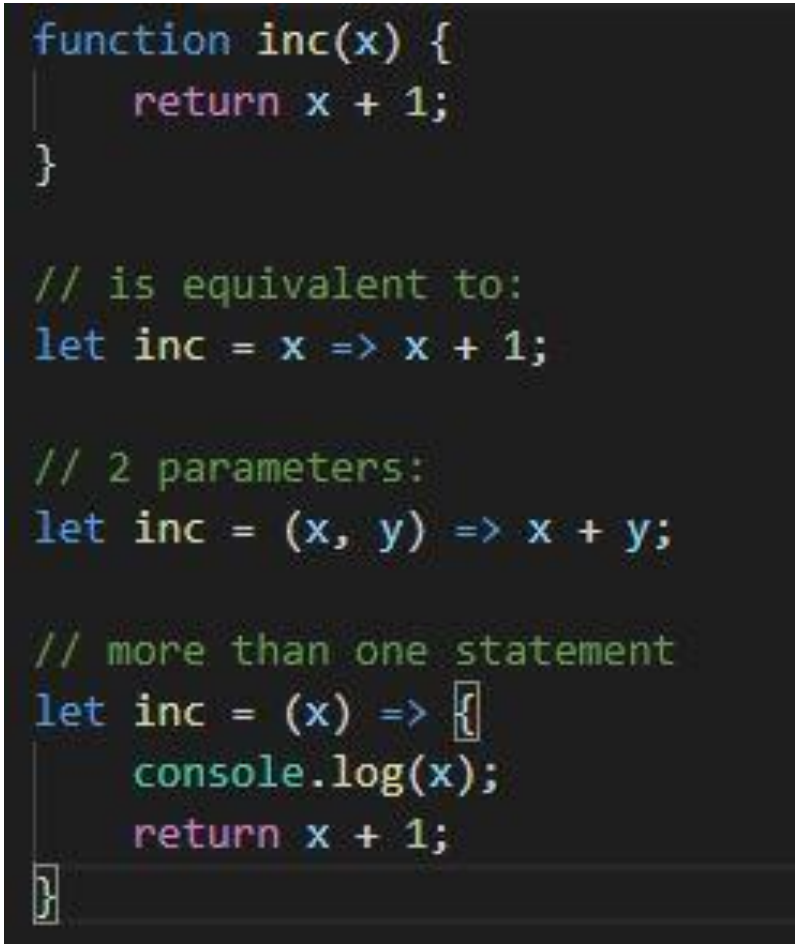
```
const bar = 1;
bar = 100; // Error!
const bar = 1000; // Error!

// properties are mutable
const obj = {};
obj.foo = 1; // No error
```

Arrow Functions

Arrow functions

- Fat arrow notation => used



Arrow functions

- Capture the ‘this’ value of the enclosing context

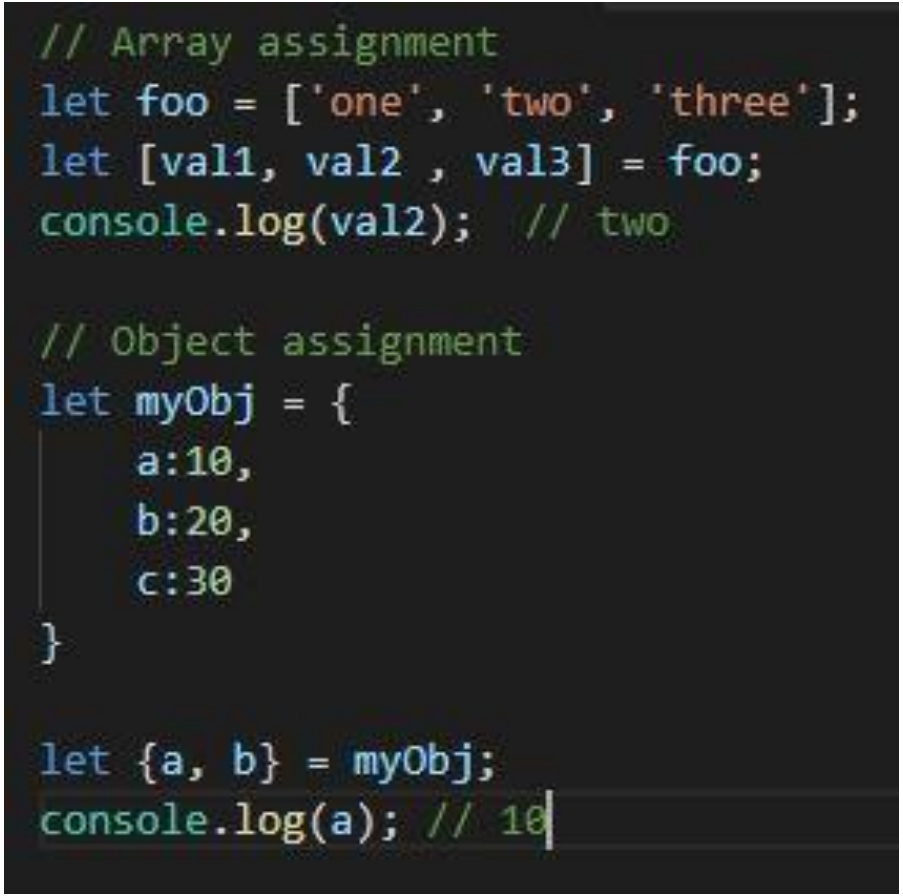
```
let obj = {
    name: 'Bob',
    friends: ['John', 'Samuel'],
    printFriends: function(){
        this.friends.forEach(f => {
            console.log(this.name + " knows "+ f);
        });
    }
}

obj.printFriends();
// Bob knows John
// Bob knows Samuel
```

Destructuring

Destructuring

- Destructuring is a way to quickly extract data out of an {} or [].



```
// Array assignment
let foo = ['one', 'two', 'three'];
let [val1, val2 , val3] = foo;
console.log(val2);  // two

// Object assignment
let myObj = {
    a:10,
    b:20,
    c:30
}

let {a, b} = myObj;
console.log(a); // 10
```

Classes and Inheritance

Classes and Inheritance

- Classes support prototype-based inheritance, super calls, instance and static methods and constructors.
- ES6 introduced a new set of keywords include class, constructor, static, extends, and super.

```
class Person{
    constructor(name) {
        this.personName = name;
    }
    move(distance = 0) {
        console.log(`${this.personName} moved ${distance}m.`);
    }
}

class AnotherPerson extends Person {
    constructor(name, age) {
        super(name);
        this.age = age;
    }
    getDetails(){
        console.log(`${this.personName}'s age is ${this.age}.`);
    }
}

let p1 = new AnotherPerson('Ajay', 22);
p1.move(100);       // Ajay moved 100m.
p1.getDetails();    // Ajay's age is 22.
```

Modules

Modules

- A javascript file is a module
- **exports** - simply means public
- **import** - simply means referencing a code from javascript file/ module

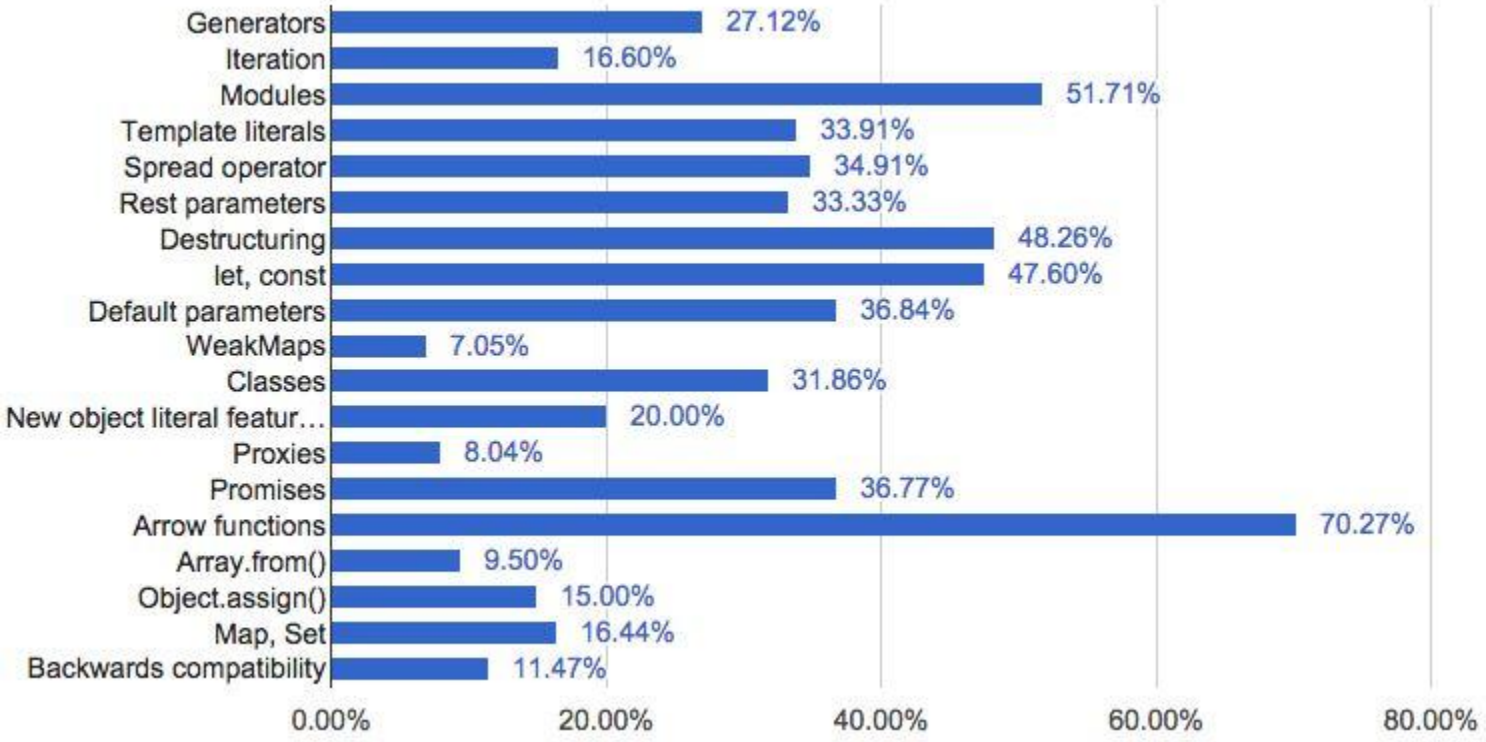
```
// lib/math.js
export function sum(x, y) {
    return x + y;
}

export var PI = 3.141593;
```

```
// app.js
import {sum, PI} from "./lib/math";

console.log("2PI = "+ sum(PI, PI));
```

Most Popular ES6 Feature: Arrow Functions



TypeScript

Introduction

- TypeScript is open source project maintained by Microsoft.
- TypeScript is a language for large scale JavaScript application development.
- TypeScript is typed superset of JavaScript that compiles to plane JavaScript. (ES3/ES5)
- TypeScript adds **static typing, classes, modules** to JavaScript.
- TypeScript can not run on any browser, any host, any OS on its own. (Need to transpile the code using babel or any other transpiler)
- TypeScript 1.0 released in April 2014, and purposefully borrows ideas from ES6 (EcmaScript 6.0).

Features

Features

- Optional Static Type Annotation / Static Typing
- Additional Features for Functions
  - Types for function parameters and return type, optional and default parameter, rest parameter
- Class
  - Field, Property, Method, Constructor, Event, Static Methods, Inheritance
- Interface
- Module
- Generics
- Few other features...

Types / Optional Type Annotation

Optional Static Types

- Any
- Primitive
  - Number
  - Boolean
  - String
  - Void
  - Null
  - Undefined
    - same as javascript “undefined” type
- Array
- Enum

```
let isDone: boolean = false;
let height: number = 6;
let name: string = "bob";
let list: number[] = [1, 2, 3];
let list: Array<number> = [1, 2, 3];

enum Color {Red, Green, Blue};
let c: Color = Color.Green;

let notSure: any = 4;
notSure = "maybe a string instead";
notSure = false; // okay, definitely a boolean

function showMessage(data: string): void {
  alert(data);
}
```

Compiling TypeScript source file

Install the TypeScript transpiler using npm:

$ npm install -g typescript

Then use tsc to manually compile a TypeScript source file into ES5:

$ tsc test.ts

$ node test.js

Interface

Interface

- An interface is an abstract type, it does not contain any code as a class does.
- It only defines the 'signature' or shape of an API.
- It is very similar to ‘interface’ in Java programming language.

```
interface Employee {
  name:string
  basic:number
  allowance:number
}

function getEmpSalary(emp:Employee):number {
  return emp.basic + emp.allowance;
}

var emp1 = {name:"ABC", basic:100, allowance: 30}

console.log(getEmpSalary(emp1));

// commands
// tsc emp.ts
// node emp.js
```

References

ES6:

http://es6-features.org/#Constants

TypeScript:

https://www.typescriptlang.org/docs/home.html

Thank You!