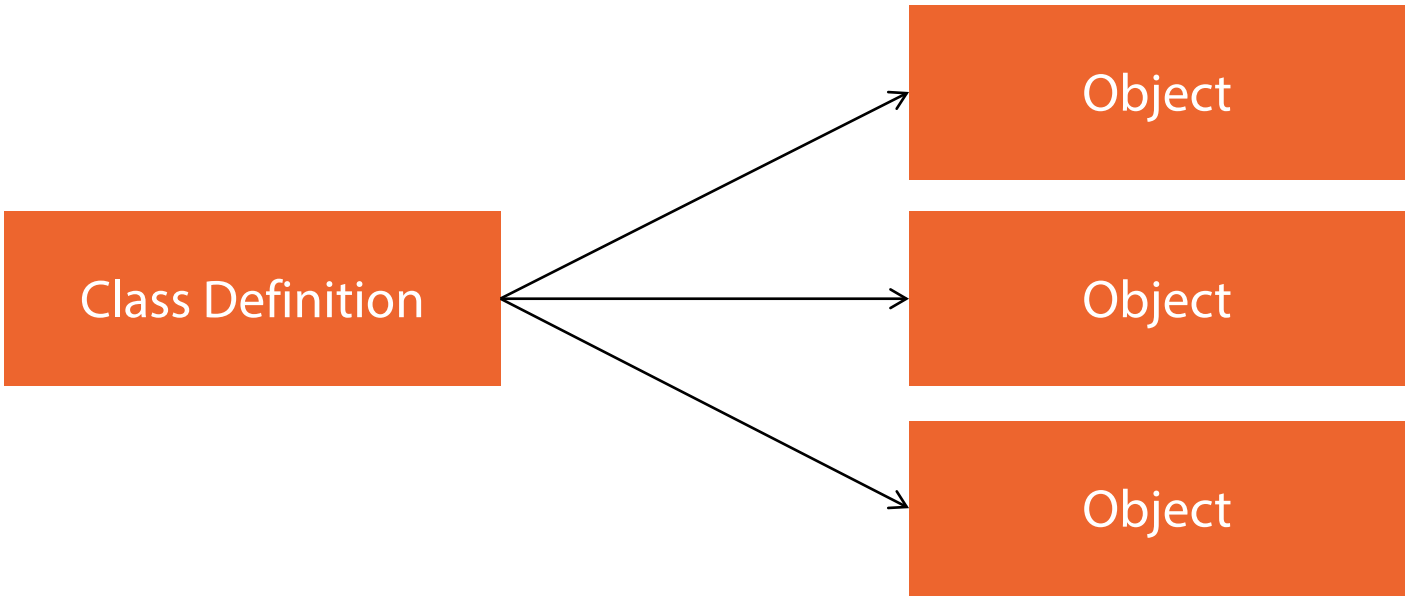
Classes

ECMAScript 6 Fundamentals

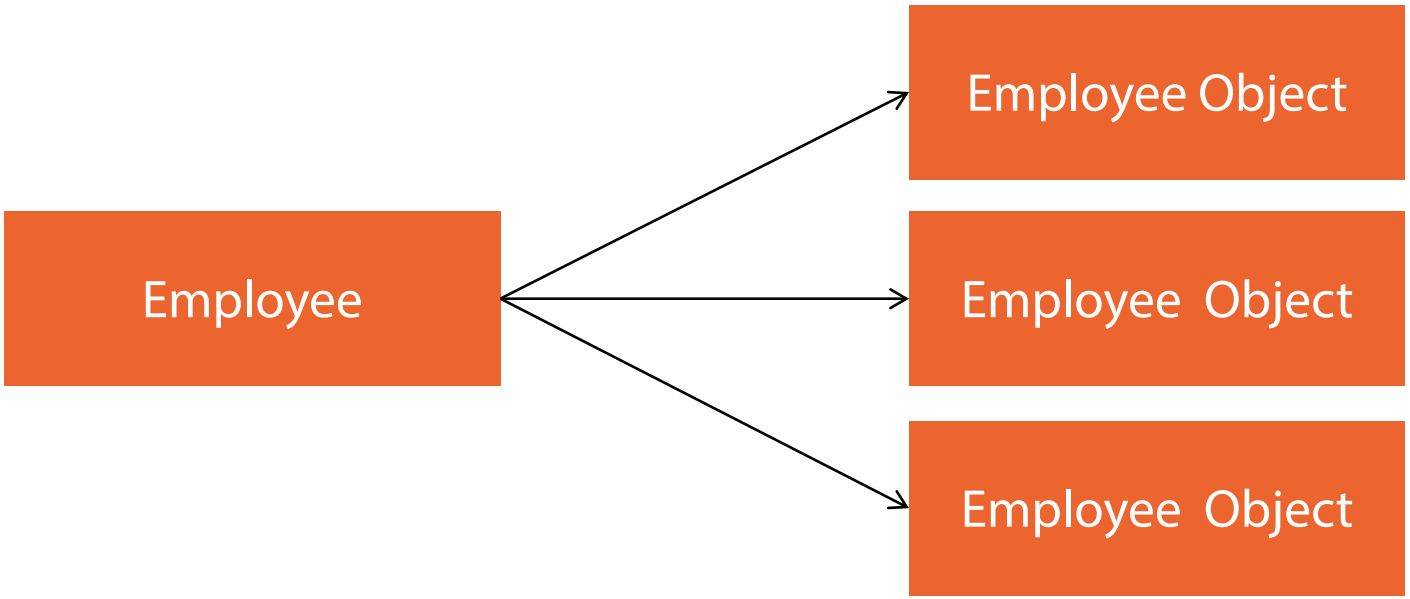
K. Scott Allen
odetocode.com
@OdeToCode

pluralsight
hardcore dev and IT training

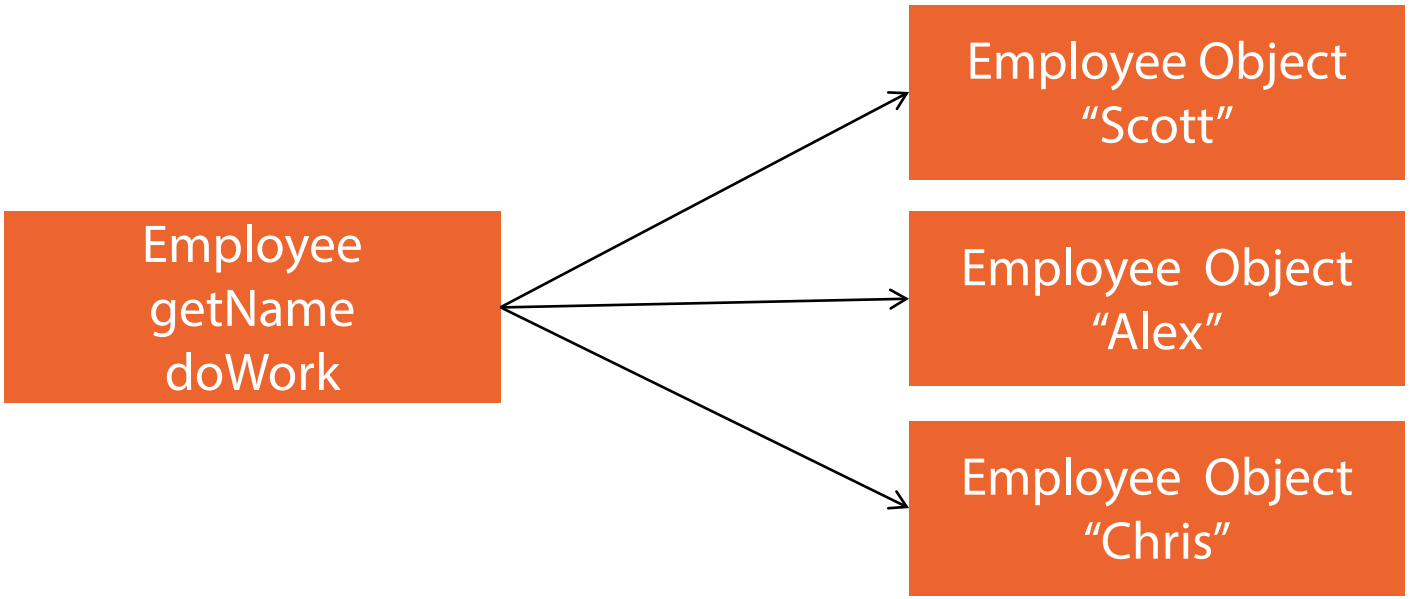
Why Classes?



Why Classes?


Employee
Employee Object
Employee Object
Employee Object

Why Classes?


Employee
getName
doWork
Employee Object
“Scott”
Employee Object
“Alex”
Employee Object
“Chris”

Class versus Prototype

```
var Employee = function() {

};

Employee.prototype = {
    doWork: function(){
        return "complete!";
    }
};

var e = new Employee();
expect(e.doWork()).toBe("complete");
```

```
class Employee {
    doWork() {
        return "complete!";
    }
}

var e = new Employee();
expect(e.doWork()).toBe("complete!");
```

constructor

```
class Employee {

    constructor() {

    }

    doWork() {
        return "complete!";
    }

    getName() {
        return "Scott";
    }
}
```

Getters and Setters

```
class Employee {

    get name() {
        return "...";
    }

    set name(newValue) {
        // ....
    }

}
```

Inheritance

```
class Person {

    constructor(name) {
        this.name = name;
    }

    get name() {
        return this._name;
    }

    set name(newValue) {
        this._name = newValue;
    }
}
```

```
class Employee extends Person {
    // employee "is-a" person
}
```

Employee is a Person

Circle is a Shape

Car is a Vehicle

Car is an Engine

(No, Car "has" an Engine)


Person
Employee
(is a Person)
Manager
(is an Employee)
Executive
(is a Manager)

super

```
class Employee extends Person {
    constructor(name, title){
        // name ??
        this._title = title;
    }
}
```

```
class Employee extends Person {
    constructor(name, title){
        super(name);
        this._title = title;
    }
}
```

Conclusion

```
class Employee extends Person {

    constructor(title, name) {
        super(name);
        this._title = title;
    }

    get title() {
        return this._title;
    }

    doWork() {
        return `${this._name} is working`;
    }

}
```