Chapter6 Notes

Methods

```
let rabbit = {};
rabbit.speak = function(line) {
    console.log(`The rabbit says '${line}'`);
};
rabbit.speak("I'm alive.");
```

instance-object

```
class Dog
{
    age: number
    breed: string
    constructor(age: number, breed: string)
    {
        this.age = age
        this.breed = string
    }
    getRelativeAge(): number
    {
        return this.age * 7
    }
}
let Spot = new Dog(2, 'Labrador')
```

1. Equivalent function in **ES5**

```
function Dog(age, breed)
{
    this.age = age
    this.breed = breed
}
Dog.prototype.getRelativeAge = function() {
    return this.age * 7
}
var Spot = new Dog(2, 'Labrador')
```

Inheritance

```
class Animal
{
    age: number
    breed: string
    constructor(age: number, breed: string)
    {
        this.age = age
        this.breed = breed
    }
    makeSound_(sound: string): void
    {
        console.log(sound)
        console.log(sound)
        console.log(sound)
    }
}
```

Using super

```
class Dog extends Animal
{
    playsFetch: boolean
    constructor(age: number, breed: string, playsFetch: boolean)
    {
         super(age, breed) // call parent constructor
         this.playsFetch = playsFetch
    }
    makeSound(): void
    {
        super.makeSound_('woof woof')
    }
    getAgeInHumanYears(): number
    {
        return this.age * 7    // super.age will throw error
    }
}
class Cat extends Animal
{
    constructor(age: number, breed: string)
    {
        super(age, breed)
    }
    makeSound(): void
    {
        super.makeSound_('meow meow')
    }
}
```

Access Control

```
class Dog
{
    public name: string // leaving out 'public' would work too
}
class PetStore
{
    dogs: Array<Dog>
    printAllDogNames(): void
    {
        this.dogs.forEach(dog => {
            console.log(dog.name)
        })
    }
}
```

1. Getters & Setters

```
class Dog
{
    private _name: string // beginning underscore is convention
    get name(): string
    {
       return this._name
    }
    set name(name: string): void
    {
        if(!name || name.length > 20) {
            throw new Error('Name invalid')
        }
        else {
            this._name = name
        }
    }
}
class PetStore
{
    private _dogs: Array<Dog> // we changed this to private too
    constructor()
    {
        this._dogs = [new Dog()]
        this._dogs[0].name = 'Fido' // will call 'set'
    }
    printAllDogNames(): void
    {
```

```
        this._dogs.forEach(dog => {
            console.log(dog.name) // will call 'get'
        })
    }
}
```

1. protected

```
class Animal
{
    protected makeSound_(sound: string): void
    {
        console.log(sound)
        console.log(sound)
        console.log(sound)
    }
}
class Dog extends Animal
{
    makeSound(): void
    {
        super.makeSound_('woof woof')
    }
}
class PetStore
{
    makeSomeSounds(): void
    {
        let dog = new Dog()
        dog.makeSound() // => 'woof woof' 'woof woof' 'woof woof'
        let animal = new Animal()
        animal.makeSound_() // => NOT ALLOWED
    }
}
```

Other Modifiers

1. static

```
class Dog
{
    static species = 'Canis Familaris'
    age = 10
}
class PetStore
{
    printSpecies(): void
```

```
    {
        console.log(Dog.species) // => 'Canis Familaris'
        console.log(Dog.age) // => undefined
    }
}
```

1. readonly

```
class Dog
{
    static readonly species = 'Canis Familaris'
}
class PetStore
{
    printSpecies(): void
    {
        console.log(Dog.species) // => 'Canis Familaris'
        Dog.species = 'Terdus Maximus' // => NOT ALLOWED
    }
}
```

Interfaces

```
interface iDog
{
    getPedigree: Function
}
class Dog implements iDog
{
    getPedigree(): Promise<Pedigree>
    {
        return someThirdPartyIoCall('...')
    }
}
class MockDog implements iDog
{
    getPedigree(): Promise<Pedigree>
    {
        return new DummyPedigreeObject()
    }
}
async function methodToBeTested(dog: iDog): Promise<void>
{
    try {
        let pedigree = await dog.getPedigree()
        // do assertions here
```

```
    }
    catch(err) {
        console.log(err)
    }
}

// Real World
methodToBeTested(new Dog())

// During Testing
methodToBeTested(new MockDog())
```

Abstract

```
abstract class Animal
{
    protected age_: number
    abstract getRelativeAge(): number;
}
class Dog extends Animal
{
    getRelativeAge(): number
    {
        return this.age_ * 7
    }
}
class Cat extends Animal
{
    getRelativeAge(): number
    {
        return this.age_ * 6
    }
}
```