```
/*
* constructor function example
 */
/* const Dice = function(sides=6) { // defaults to 6 if nothing is provided by user
    this.sides = sides;
    this.roll = function() {
        return Math.floor(this.sides * Math.random() + 1);
    }
}
*/
/*
* instance of Dice
 */
/*
const redDie = new Dice(); // this will be a six-sided die - normal cube
const whiteDie = new Dice(4); // four sided die - 3-sided pyramid
const blueDie = new Dice(5); // five sided die - can a five sided die be fair?
*/
/*
* using the class declaration
* This was added in ES6
 */
class Dice {
    constructor(sides=6){
        Object.defineProperty(this, 'sides', {
            get() {
                return `This dice has ${sides} sides`;
            },
            set(value) {
                if(value > 0) {
                    sides = value;
                    return sides;
                } else {
                    throw new Error('The number of sides must be positive');
                }
            }
        });
        this.roll = function() {
            return Math.floor(sides * Math.random() + 1)
        }
    }
    static description() {  // static functions are not accessible from instances of
    the class
        return 'A way of choosing random numbers'; // only accessible from the class
        itself
        /* Dice.description(); */
    }
}

const redDie = new Dice(); // creates a six-sided die
const blueDie = new Dice(4); // creates a four-sided die
const whiteDie = new Dice(5); // creates a five-sided die

let fourSide = whiteDie.roll();

console.log(Dice.description());

/*
* example of prototype properties and methods
 */

class Turtle {
    constructor(name) {
        this.name = name;
        this.weapon = 'hands';
    }
```

```
65
66      sayHi() {
67          return `Hi dude, my name is ${this.name}`;
68      }
69      attack() {
70          return `Feel the power of my ${this.weapon}`;
71      }
72  }
73
74  const leo = new Turtle('Leonardo'); // leo is not an instance of the Turtle class
75  // it has a name property and sayHi method references the name property
76
77  leo.sayHi(); // returns 'Hi dude, my name is Leonardo'
78  leo.attack(); // returns 'Feel the power f my hands'
79
80  class NinjaTurtle extends Turtle {
81      constructor(name) {
82          super(name);
83          this.weapon = 'hands';
84      }
85      attack() {
86          return super.attack();
87      }
88  }
89
90  /*
91  * add isEven() and isOdd() to the Number wrapper object's prototype
92   */
93
94  Number.prototype.isEven = () => this % 2 === 0;
95  Number.prototype.isOdd = () => this % 2 === 1;
96
97  /*
98  * mixin: change the built-in function to make a deep copy, rather than a by reference
    copy
99   */
100 function mixin(target,...objects) { // first parameter: is the object we are applying
    the mixin to
101     for (const object of objects) {
102         if(typeof object === 'object') {
103             for (const key of Object.keys(object)) {
104                 if (typeof object[key] === 'object') {
105                     target[key] = Array.isArray(object[key]) ? [] : {};
106                     mixin(target[key],object[key]);
107                 } else {
108                     Object.assign(target,object);
109                 }
110             }
111         }
112     }
113     return target;
114 }
```