JavaScript 高级

- 编程思想：
  - 面向过程：
    - 什么是面向过程？ 面向过程就是分析出解决问题所需要的步骤，然后用函数把这些步骤一步一步实现，使用的时候再一一个个的依次调用就可以了
    - 优缺点：
      - 优点：性能比面向对象高，步骤练习紧密
      - 缺点：不好维护，不易多次使用及扩展
  - 面向对象：
    - 什么是面向对象？ 面向对象是把事务分解成为一个个对象，然后由对象之间分工与合作
    - 三大特性：
      - 1.封装性
      - 2.继承性
      - 3.多态性
    - 优缺点：
      - 优点：易维护，可复用，可扩展，灵活性高
      - 缺点：性能没有面向过程高
- 类和对象
  - ES6
    - 对象 什么是对象？ 对象是具体的
    - 类
      - 什么是类？ 类模拟抽象的，泛指的
      - 创建类 例：class 类名 { } 类名首字母要大些
      - constructor构造函数 例：class Star { constructor (uname,age){ this.uname = uname; this.age = age; } }
        - 注意：1.类里面的方法不带function，直接写既可
        - 2.类中定义属性，调用方法都得用this
        - 3.方法之间不能加逗号分隔
      - 继承： extends
      - super关键字： 用于访问和调用对象父类上的函数 注意：子类在构造函数中使用super, 必须放到this 前面(必须先调用父类的构造方法，在使用子类构造方法）
      - 三个比较重要的注意点：
        - 在ES6中类没有变量提升，所以必须先定义类，才能通过类实例化对象.
        - 类里面的共有属性和方法一定要加this使用.
        - this指向的问题
          - 方法里面的this，谁调用，this就代表谁
          - 构造方法里面的this，代表当前实例对象
  - ES5
    - 成员
      - 1.静态成员 不需要实例对象调用属性（不需要new）
      - 2.实例成员 需要实例对象调用属性（需要new）
    - 构造函数原型（prototype 原型对象-构造函数的一个属性）
      - 1.所有的属性都要写在构造函数里面
      - 2.所有的方法都要写在原型对象里面
    - 对象原型:简称"原型" （__proto__）
      - 1.每一个对象都有一个原型
      - 2.作用是指向原型对象prototype
    - 构造函数 constructor 作用：指回原来的构造函数
    - 组成原型链：作用：提供一个查找成员的机制
    - 继承：
      - 属性继承 call(this,m,n)--------m,n为不确定参数，参数个数可变
      - 方法继承
        - 原理：1.子类的原型对象 《=》父类的实例对象
        - 2.继承完毕之后，子类的constructor的指向已经不见，需要将子类的custructor手动指回
- 函数
  - 声明方式
    - 1.命名函数 function fn（）{}
    - 2.匿名函数 var fn = function(){}
    - 3.new function函数 var fn = new Function('a', 'b', 'console.log(a + b)')
  - 调用：
    - 1.普通函数
      ```
      function fn() {
          console.log('人生的巅峰');
      }
      fn();
      ```
    - 2.对象的方法
      ```
      var o = {
        sayHi: function() {
          console.log('人生的巅峰');
        }
      }
      o.sayHi();
      ```
    - 3.构造函数 function Star() {}; new Star();
    - 4.绑定事件函数 btn.onclick = function(){}
    - 5.定时器函数 setInterval(function() {}, 1000); 注意：非定时器的代码要快于定时器代码（即使定时器代码时间为0）
    - 6.立即执行函数(自调用函数) (function (){ console.log('自调函数'); })();
  - this
    - 指向
      - 1.普通函数 window
      - 2.对象函数 该方法所属的对象
      - 3.构造函数 new出来的实例对象
      - 4.绑定事件函数 事件源，绑定时间对象
      - 5.定时器函数 window
      - 6.立即执行函数（自调用函数） window
    - 改变指向
      - 1.call（this，m，n.....）
        - 参数： 参数使用，逗号分隔开
        - 应用场景： 父子继承上面
      - 2.apply（this，[m，n.....]）
        - 参数： 参数使用[] 数组包含上
        - 应用场景： 常常跟数组有关系，例如：借助Math种的最大值，最小值，求出数组中的最大值，最小值
      - 3.bind（this，m，n.....）
        - 参数 参数使用，逗号分隔开
        - 应用场景 不调用函数，并且还想改变this的指向
      - 特点：调用函数
  - 模式：
    - 正常模式：
    - 严格模式：
      - 1.声明： "use strict"
      - 2.使用位置
        - 1.函数的顶端（使当前的函数声明为严格模式）
        - 2.脚本的顶端（是脚本声明为严格模式）
      - 3.注意事项
        - 1.变量
          - 1.变量声明中，必须加var
          - 2.不准删除变量
        - 2.this的指向：
          - 正常模式： 在全局的作用域下，函数中的this，指向的是window
          - 严格模式： 在全局的作用域下，函数中的this，是undefined
        - 3.函数变化
          - 1.函数不能有重复函数
          - 2.非函数体内不能含有函数体
  - 高阶函数：
    - 1.将函数当作值传入参数
    - 2.将函数当作返回值返回
  - 闭包 一个作用域访问另一个作用域内的局部变量
- 递归
  - 什么是？ 函数调用函数本身
  - 深拷贝与浅拷贝
    - 浅拷贝 第一层：简单数据类型 扩展：ES6新增方法Object.assign("新的"，"源的")
    - 深拷贝 实现复杂类型的赋值，切记不能直接赋值，考虑复杂类型地址的问题
  - 补充知识： instenceof 判断是不是数组或者对象
- 正则表达式
  - 什么是？ 匹配字符串、检索、验证，表单验证
  - 创建方式
    - 1.var reg = new RegExp(/abc/)
    - 2.var reg = /abc/
    - 这两个创建完毕的意思字符串中是否含有abc，返回boolean
  - 组成 简单字符和特殊字符【元字符】
  - 边界符
    - n表示匹配行首的文本
    - $表示匹配行尾的文本
  - 中括号
    - 例：var reg = /^[abc]$/ 也可以写成 var reg = /^[a|b|c]$/
      - a或b或c
      - 多选1
    - ^ 这个需要注意一下：括号外边是边界符，里面是取反的意思
  - 量词符
    - * 重复0次或者多次
    - · 任意的
    - + 最少一次
    - ? 重复0次或者1次
    - {n} 重复n次
    - {n，} 重复n次或者更多次
    - {n，m} 重复n到m次
  - 括号总结
    - 大括号 量词符 表示重复里面的次数
    - 中括号 匹配括号中的任意字符
    - 小括号 表示优先级