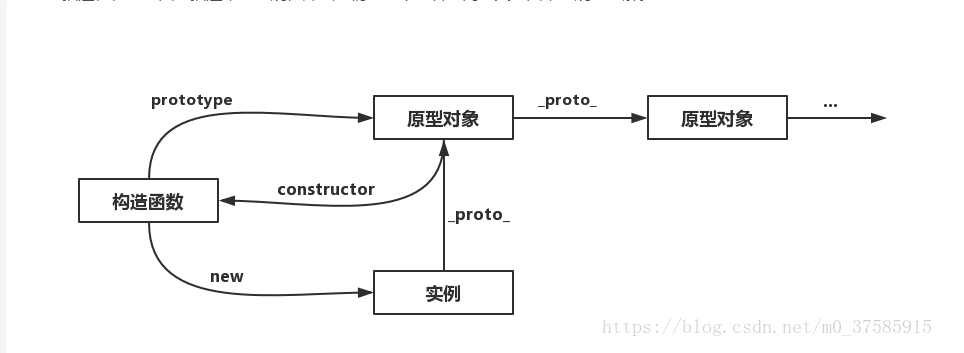
1. 原型链



有构造函数F(),new出实例f = new F()

构造函数F.prototype等于实例f.\_\_proto\_\_,原型对象的constructor即F.prototype.constructor 等于构造函数F，原型链可沿着\_\_proto\_\_一直找到Object.prototype。

1. 闭包

https://blog.csdn.net/weixin\_43558749/article/details/90905723

1. **闭包有3个特性：  
   ①函数嵌套函数  
   ②函数内部可以引用函数外部的参数和变量  
   ③参数和变量不会被垃圾回收机制回收**
2. **闭包的缺点：可能导致内存泄漏；函数使用同一个变量，导致意外的结果。**
3. **解决办法：在外部进行封装，把变量当初参数传递，创造一份独立的数据。**
4. **闭包的优点：可以用作缓存，存储数据**
5. 基本数据类型

https://blog.csdn.net/weixin\_39786171/article/details/80699736

基本数据类型：number,string,null,bool,undefined,Symbol

引用数据类型：Function,Object,Array,Date

1. 基本数据存放在栈内存，引用数据存放在堆内存，但引用类型的地址存放在栈中，可以通过地址访问堆中的数据
2. 基本数据类型的赋值是值赋值（创建一份新的副本，数据a、b以后的赋值不会相互影响），引用数据类型的赋值是引用赋值（对象a、b用的同一个地址，指向同一份数据，一个改变会影响到另一个）
3. 基本数据类型的比较是值的比较（即a、b的值相同即可），引用数据类型的比较是引用地址的比较（即使a、b的值相等，但地址不同，也是不相等的）
4. 任何方法都无法改变一个基本类型的值

a=”hello”;

a.toUpperCase();//输出HELLO

console.log(a)//输出hello

1. 基本数据类型无法添加方法和属性

a=”hello”

a.sayHello=functiong(){}

a.name=”yhc”

console.log(a. sayHello)//输出undefined

console.log(a.name) //输出undefined

1. bind,apply,call

相同点: 改变this的指向

不同点：bind返回的是函数，传递的参数是绑定对象（即新的this）和数组；call返回结果，传递的参数是绑定对象（即新的this）和数组；apply返回结果，传递的参数是绑定对象（即新的this）和各个函数所需的参数，用逗号分隔，如fun.apply(obj, a, b, c)

1. Promise

let promise = new Promise((resolve, reject) => {

if(xxx){

resolve(res)

}else{

reject(new Error(err))

}

})

promise.then(res =>{

return new Promise()//若是要链式调用，可在此处new Promise(),不需要则不return

}, err=>{})

1. async

async doxxx(){

let res1= await promise1()

let res2 = await promise2(res1)//链式调用，使用前一个的结果，会等待前一个执行完毕再执行

return [res1, res2]//两个同时成功才会触发then的resolve

}

doxxx().then(res=>{},err=>{})

1. eventLoop

浏览器是单线程，eventLoop里有宏任务、微任务、主线程（同步任务）。

1. 浏览器逐行执行代码，遇到同步任务立即执行，遇到微任务丢进微任务队列，遇到宏任务丢进宏任务队列
2. 检查微任务队列，若队列不为空，执行微任务，跳到1执行；若为空，跳到步骤3
3. 检查宏任务队列，若队列不为空，执行宏任务，跳到1执行；若为空，结束事件循环

同步任务：各位立即执行函数，new Promise(),console.log()

微任务：Promise.then()

宏任务：鼠标事件，定时函数，io操作，ui rendering