**Js中同步和异步问题**

**单进程：当前运行的整个程序**

**多进程：同时运行多个软件和程序**

**单线程：意味着需要排队。前一个任务结束后，才执行下一个任 务，会拖延整个程序的执行。常见的浏览器无响应**

**多线程：**

**同步任务：在主线程上排队执行的任务，只有前一个任务执行完毕，才能执行后一个任务**

**异步任务：不进入主线程，而进入“任务队列”的任务，只有**

**“ 任务队列”通知主线程，某个异步任务可以执行了， 该任务才 会进入主线程执行**

**事件监听：采用事件驱动模式**

**任务的执行不取决代码的顺序，而取决于某一个事 件是否发生**

**事件轮询机制：只要主线程空了，就会读取“任务队列”，这就是javascript的运行机制。这个过程会不断重复，称为“事件轮询机制”**

**异步执行的运行机制：**

**1、所有的同步任务都在主线程上执行,形成一个执行栈(execution context stack)**

**2、主线程之外，,还有存在一个“任务队列” (task queue).只要异步任务有了运行结果，就在“任务队列” 之中放置一个事件**

**3、一旦“执行栈”中的所有同步任务执行完毕,系统就会读取“任务队列”, 看看里面有那些事件,那些对应的异步任务,于是结束等待状态,进入执行栈, 开始执行**

**4、主线程不断重复上面的第三步**

**常见的异步任务：**

**回调函数不一定是异步，在异步操作中的回调 Setinterval**

**settimeout**

**ajax**

**事件监听： on、bing、listen、 addeventlistener、**

**Observe**

**发布/订阅**

**Promise对象**

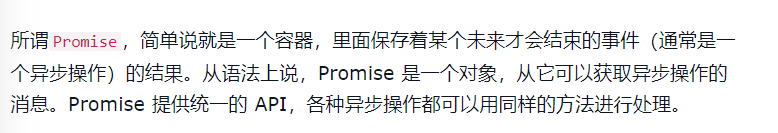
**Generator 函数**

**Async函数**

**API:提供了统一的接口，使异步操作更加容易，解决了通过回调函数处理异步操作的回调函数的地狱问题。**

**Promise含义**

promise 是异步编程的一种解决方案，比传统的解决方案——回调函数和事件——更合理和更强大。



**Promise的特点**

**对象的状态不受外界影响**

**Promise对象代表一个异步操作，有三种状态**

**进行中(pending)**

**已成功(fulfilled)**

**已失败(rejected)**

**只有异步操作的结果，才能决定promise的状态**

**一旦状态改变，就不会再改变。**

**就是一旦已定性(resolved)，再向promise对象中添 加回调函数，就会立即得到这个结果**

**优点：可以将异步操作以同步操作的流程表达出来，**

**Promise对象提供统一的接口，使得控制异步操作 更加容易**

**缺点：一旦建立它就会执行，无法取消**

**如果不设置回调函数，promise内部就会抛出错误**

**不会得到外部响应**

**当处于pending状态使，无法得知目前的进行到 那一阶段**

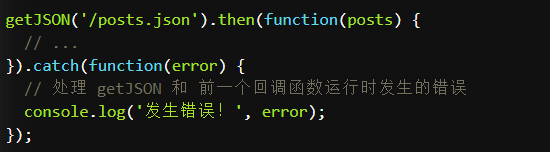
**Promise方法**

**Promise.prototype.then() //第一个参数定义成功**

**//第二个参数定义失败**

**Promise.prototype.catch()**

**//捕获promise错误，内部错误也能捕获**



**Promise.prototype.finally() //用于指定不管promise 对象状态如何，都会执行**

**不接受任何参数**

**Promise.all() 用于将多个 Promise 实例，包装成一个新 的 Promise 实例，参数是一个数组**



**Promise.race() 和Promise.all() 相同**

**区别：race 只能得到第一个状态发生改变的**

**All 当中必须都成功才能返回**

**Promise.allSettled() 只会成功fulfilled 不会失败**

**Promise.any() 一个成功合并就成功，所有失败才失败**

**Promise.resolve() 转换**

**Promise.reject()**

**Iterator和for..of循环**

**遍历器：是一种接口，是一种机制，提供了通统一的遍历方法**

**作用：**

**1、为各种数据结构，提供一个统一的、简便的访问接口**

**2、使得数据结构的成员能够按某种次序排列**

**3、ES6创造了一种新的遍历命令for..of循环，lterator接 口主要供for..of消费**

**Iterator接口的目的：就是为所有数据结构，提供了一种统一 的访问机制**

**数据结构只要部署了lterator接口，就可以完成便利操作**

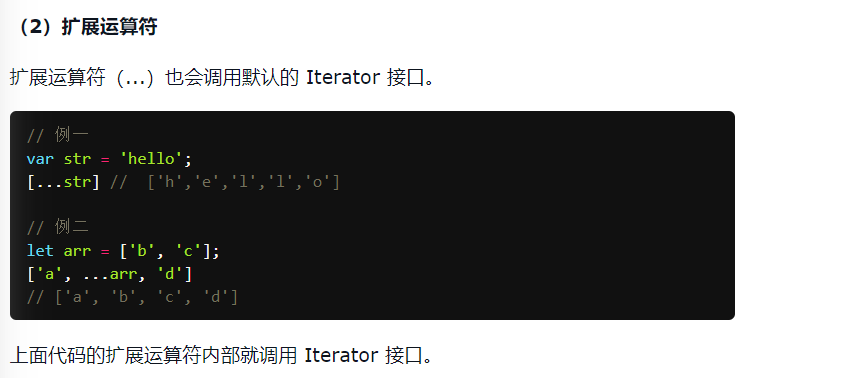
**注：自己创建的对象是没有Iterator对象的**

**原生具备Iterator接口的数据结构：五个**

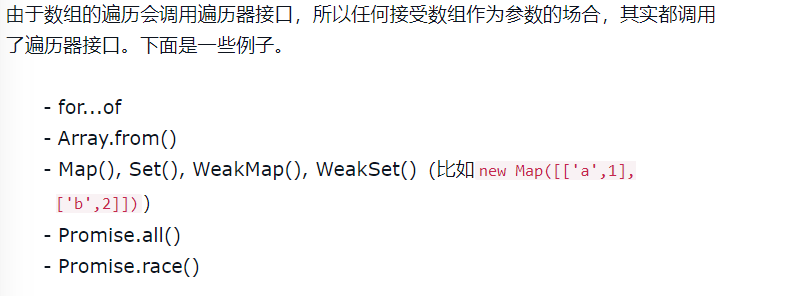
**Array,Map,Set,String,**

**TypedArray类数组：函数的arguments对象NodeList对象**









**Generator 函数**

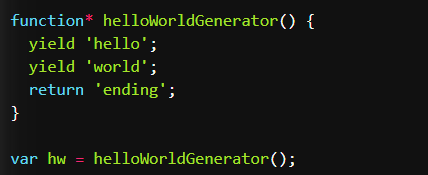
**Generator 函数是ES6提供的一种异步编程解决方案，语法行为与传统函数 完全不一样**

1. **可以封装对多个状态**

**Function关键字与函数名之间有一个 星号\***

**函数体内部使用yield表达式，定义不同的内部状态**

**语法：**



Console.log(iter.next())

**Next()方法**

**yield表达式是没有返回值的，或者说返回undefined**

**Next的参数可以作为上一个yield表达式的返回值**

**应用：**

**异步操作的同步化表达**

**Promise和generator的区别**

**Promise的写法只是回调函数的改进，使用then方法以后，异步任务的两段执行看的更清楚了，除此之外，别无新意**

**Promise最大的问题是代码冗余，原来的任务被promise包装了一下，不管什么操作，一眼看去都是一堆then，原来的语义变得很不清楚**

**async函数**



1. **内置执行器**
2. **更好的语义**
3. **更好的适用性**
4. **返回值是prmise**

