1. 红宝书 dom扩展
2. Promise
   1. Promise对象代表一个异步操作，有3种状态：Pending进行中、Fulfilled已成功、Rejected已失败。

Promise的状态改变只有2种可能：从进行中到成功、从进行中到失败。一旦状态改变就不会再变。

无法取消Promise，一旦新建就会立即执行。

* 1. 例子1

/\*

Promise对象是一个构造函数，用来生成Promise实例

Promise实例生成后，可以用then 方法指定Resolved 状态和Rejected状态的回调函数

\*/

function timeout(ms){

return new Promise((resolve,reject)=>{

setTimeout(resolve,ms,'done');

})

}

timeout(100).then(value=>{

console.log(value);//done

})

* 1. 例2

/\*

Promise新建后就会立即执行，

then方法指定的回调函数将在当前脚本中的所有同步任务执行完成后才会执行

\*/

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

console.log('Promise');

resolve();

})

promise.then(function(){

console.log('Resolved');

})

console.log('hi');

//结果：Promise hi Resolved

* 1. 例3 异步加载图片

function loadImageAsync(url){

return new Promise(function(resolve,reject){

var image=new Image();

image.onload=function(){

resolve(image);

}

image.onerror=function(){

reject(new Error('Could not load image at '+url));

}

image.src=url;

})

}

* 1. Promise.prototype.then()

Promise实例有then方法，即then方法是定义在原型对象Promise.prototype上的。

作用是为Promise实例添加状态改变时的回调函数。

then返回的是一个新对象，可以用链式写法

* 1. Promise.prototype.catch()

Promise.prototype.catch()是.then(null,rejection)的别名，用于指定发生错误时的回调函数。

最好用catch，不要用then中的reject的回调函数

1. 让input不能获取焦点

onfouce=”this.blur()”

1. readOnly和disabled

readOnly只对input和textarea有效

表单元素使用disabled时，表单提交时不能被传递出去

disabled时，不能获取焦点，用户的所有操作都无效