ES6

1.let语法

let是什么？

let用来声明变量,它的用法类似于var，但是所声明的变量，只在let命令所在的代码块内有效,类似于C，C++,JAVA局部变量的概念。

**基本用法：**

a.块及作用域

{

    var a = 10;

    let b = 123;

}

console.log(a);

console.log(b);

上面代码在代码块之中，分别用let和var声明了两个变量。然后在代码块之外调用这两个变量，结果let声明的变量无法被打印，var声明的变量返回了正确的值。

这表明，let声明的变量只在它所在的代码块有效。

for(let i=0; i<10; i++){

    document.write(i);

}

document.write(i)；//x 看不到

上面代码中，计数器i只在for循环体内有效，在循环体外引用就会报错。

b.暂时性死区

下面代码在有两个同名变量a，但是却由于{}使得作用域不同，输出结果为123,456。在被let修饰的同名变量下,根据就近原则，内部变量屏蔽外部变量。

let a = 456;

{

    let a = 123;

    console.log(a);

}

console.log(a);

c.不存在变量提升

var命令会发生”变量提升“现象，即变量可以在声明之前使用，值为undefined。这种现象多多少少是有些奇怪的，按照一般的逻辑，变量应该在声明语句之后才可以使用。

// var 的情况

console.log(foo); // 输出undefined

var foo = 2;

// let 的情况

console.log(bar); // 报错ReferenceError

let bar = 2;

也就意味着被let声明的变量，必须先定义在使用。

d.不允许重复声明

var a = 123;

var a = 123; //  可以实现

let b = 123;

let b = 123; //  Identifier 'a' has already been declared

2.this与bind

this:a.当this出现在普通方法时，this代表调用当前方法的对象本身。

     b.当this出现在事件中时，this代表调用当前事件的HTML元素本身。

常规函数

    function fun(){

        console.log(this);

    }

    fun();

匿名函数

    setTimeout(function(){

        console.log(this);

    },2000);

事件

    document.onclick = function(){

        console.log(this);

    }

bind(this重新指代的对象): 只能为匿名函数服务

    #box{

                width:    100px;

                height: 100px;

                background: #FF0000;

            }

        </style>

    </head>

    <body>

        <div id="box">

        </div>

    </body>

var box = document.getElementById("box");

    document.onclick = function(){

        this.style.display = "none";

    }.bind(box);

有名函数重新指定this对象: apply  call

function fun(){

        console.log(this);

    };

//    fun.apply(box);

//    fun.call(box);

fun();

3.JSON.parse/JSON.stringify

JSON字符串和对象直接的转换:

var str = '{"id":"123","name":"大黄"}';

var obj = JSON.parse(str);

document.write(obj.id  + " " + obj.name);

var obj1 = {"id":"123","name":"大黄"};

var str1 = JSON.stringify(obj1);

document.write(str);

4.const用法

什么使const命令呢？实际上它也是一种声明常量的方式。

const命令用来声明常量，一旦声明，其值就不能改变。也就是只读变量。

    const a=10;

    a=100; // Assignment to constant variable.  报错

**基本用法:**

a.声明常量

const PI = 3.1415;

PI // 3.1415

PI = 3;

// TypeError: Assignment to constant variable.

b.常量必须初始化

const foo；//报错 Missing initializer in const declaration

c.块级作用域

{

    const foo = 123;

}

console.log(foo);

上述代码依然报错，和let的效果是一样的。

d.声明不提升(const修饰的变量必须先定义后使用)

if (true) {

    console.log(MAX); // ReferenceError

    const MAX = 5;

}

e.const修饰的变量不可重复定义

简单来说，let和const的功能基本类似，只是let用来修饰变量,const用来修饰常量。

无论是let还是const,这并不算新的语法，其实这些内容甚至是其他高级语言"玩剩下的",这种改动只是将JS脚本语言更加标准化了，

可见js的语法也更加的后台化。

5.for..in...和for...of...

let strArr = ["heihei","haha","laowang"];

    for(let index in strArr){ //遍历下标

        console.log(strArr[index]);

    }

    for(let t of  strArr){ //便利值

        console.log(t);

    }

6.字符串扩展方法

判断字符串是否包含在另一个字符串中

在ES5 中使用的是 indexOf() 方法，在ES6中可以使用includes()、startsWith()、endsWith()三个方法。

str.includes(参数): 返回布尔值，表示是否找到了参数字符串。

str.startsWith(参数)：返回布尔值，参数是否在源字符串的头部。

str.endsWith(参数)： 返回布尔值，参数是否在源字符串的尾部。

var str = "yuan is an animal";

console.log(str.startsWith("is"));

console.log(str.endsWith("animal"));

console.log(str.includes("an"));

JavaScript内部，字符以UTF-16的格式储存，每个字符固定为2个字节。对于那些需要4个字节储存的字符（Unicode码点大于0xFFFF的字符），JavaScript会认为它们是两个字符。

如：var str = "𡱖"; 汉字"𡱖"的编码是21c56（需要四个字节），超过了FFFF。即超过javascript本身能表示的字符的最大范围。javascript就会把“𡱖”当作两个字符处理，

即str.length是2。而且，以前的函数，如：charAt()函数一次只能读取该汉字的其中一半。

        var str = "𡱖";   //21c56

    console.log(str.length); // 29503

    console.log(str.charAt(0)); // 尿了

    console.log(str.charCodeAt(0).toString(16));  //尿了 返回当前字符的ascii值，以16进制表示 原因：一次只能读两个字节

    console.log(str.codePointAt(0).toString(16)); //正确

    console.log("\u{21c56}"); //正确

7.箭头函数 ：针对匿名函数

定义: () => {}

有参数 :(参数1，参数2)=>{ }

带有返回值: () => {return 返回值}

let fn = () => {

        console.log("heihei");

    }

fn();

事件同样可以换成箭头函数

document.onclick = () => {

        console.log("document");

    }

setTimeout(()=>{console.log(123)},2000);

8.解构赋值:解析结构进行赋值

强调格式匹配：等号左边和右边一致

好处：

1.一次性定义很多个变量

//let x =10,y=20,z=30;

    //数组赋值

    let [x,y,z] = [10,20,30];

    console.log(x,y,z);

    //对象赋值

    let {name,age} = {name:"老王",age:18};

    console.log(name,age);

2.解构赋值可以让一个函数返回多个值   使用[]

    function fun(){

        return [1,2,3];

    }

    //let a = fun();

    let [x,y,z] = fun();

    //console.log(a);

    console.log(x,y,z);

返回1~100能被3整除的数之和，以及这样的数有多少个。

function is3(){

        let sum = 0;

        let count = 0;

        for(let i=1; i<=100; i++){

            if(i%3==0){

                sum += i;

                count++;

            }

        }

        return [sum,count];

    }

    let [sum,count] = is3();

    console.log(sum,count);

3.解构赋值可以实现两个数的交换

let a=10,b=20;

    [a,b] = [b,a];

    console.log(a,b);

冒泡排序：

let a = [6,7,8,9,0,5,4,3,2,1];

    for(let i=0; i<a.length-1; i++){

        for(let j=0; j<a.length-i-1; j++){

            if(a[j]>a[j+1]){

                [a[j],a[j+1]] = [a[j+1],a[j]];

            }

        }

    }

    for(let i=0; i<a.length; i++){

        console.log(a[i]);

    }

9.Set集合

Set集合:   var set = new Set();定义一个set集合

let set = new Set([2,3,5,7,8]);

    set.add(9);

    set.add(2);

    set.add(2);

    console.log(set);

set的遍历：

for(let t of set){

        console.log(t);

    }

Set集合特点 自动去重

实现数组去重 ：

let arr =[2,3,4,2,3,4,2,3,4];

    let set = new Set(arr);

    arr = Array.from(set);     //Number("123");

    console.log(arr);

set集合的方法：

add(参数) 向集合中添加一个元素

delete(值) 删除集合中某个数

has(值) 判断集合中是否含有某个值

clear() 清空集合

    var set = new Set([2,4,6,8,1]);

    set.add(100);

    set.delete(4);

    console.log(set.has(0));

    set.clear();

    console.log(set);

10.map ：键值对

set(key,value) 向集合中添加一个元素

get(键)  根据键去取值

delete(键) 删除集合中某个数

has(键) 判断集合中是否含有某个值

clear() 清空集合

let map = new Map();

    map.set("1","大黄");

    map.set("2","小明");

    map.set("3","老王");

    //map.delete("3");

    map.clear();

    console.log(map.has("0"));

    console.log(map.get("2"));

    console.log(map);

map的遍历:

for(let t of map){

        console.log(t[0] + t[1]); //0代表Key  1代表value

    }

11. Symbol

1、ES6引入了一种新的原始数据类型Symbol，表示独一无二的值。它是JavaScript语言的第七种数据类型，前六种是：Undefined、Null、布尔值（Boolean）、字符串（String）、数值（Number）、对象（Object）。

let symbol = Symbol();

let symbol = Symbol("heihei");主要是为了提示信息，作用不大

2、Symbol值通过Symbol函数生成。这就是说，对象的属性名现在可以有两种类型，一种是原来就有的字符串，另一种就是新增的Symbol类型。凡是属性名属于Symbol类型，就都是独一无二的，可以保证不会与其他属性名产生冲突。

let symbol1 = Symbol("heihei");

let symbol2 = Symbol("heihei");

console.log( symbol1 === symbol2 ); //false;因为每个 Symbol 值都是唯一的

3、注意，Symbol函数前不能使用new命令，否则会报错。这是因为生成的Symbol是一个原始类型的值，不是对象。也就是说，由于Symbol值不是对象，所以不能添加属性。基本上，它是一种类似于字符串的数据类型。

Symbol函数可以接受一个字符串作为参数，表示对Symbol实例的描述，主要是为了在控制台显示，或者转为字符串时，比较容易区分。

4、由于每一个Symbol值都是不相等的，这意味着Symbol值可以作为标识符，用于对象的属性名，就能保证不会出现同名的属性。这对于一个对象由多个模块构成的情况非常有用，能防止某一个键被不小心改写或覆盖。

Symbol值作为对象属性名时，不能用点运算符。在对象的内部，使用Symbol值定义属性时，Symbol值必须放在方括号之中。

symbol的属性会隐藏于for...in循环中

let age = Symbol("heihei");

    var a = {name:"老王",

             [age]:18,       看不到

             hobby:"写代码"};

    for(let t in a){

        console.log(a[t]);

    }

    console.log(a[age]);

作业

0.笔记的整理

1. 封装拖拽效果

2. 封装已学过所有兼容

3. 评分效果

4.简单版放大镜